Agile Computing and Adaptive Systems

Niranjan Suri
nsuri@ihmc.us

Outline

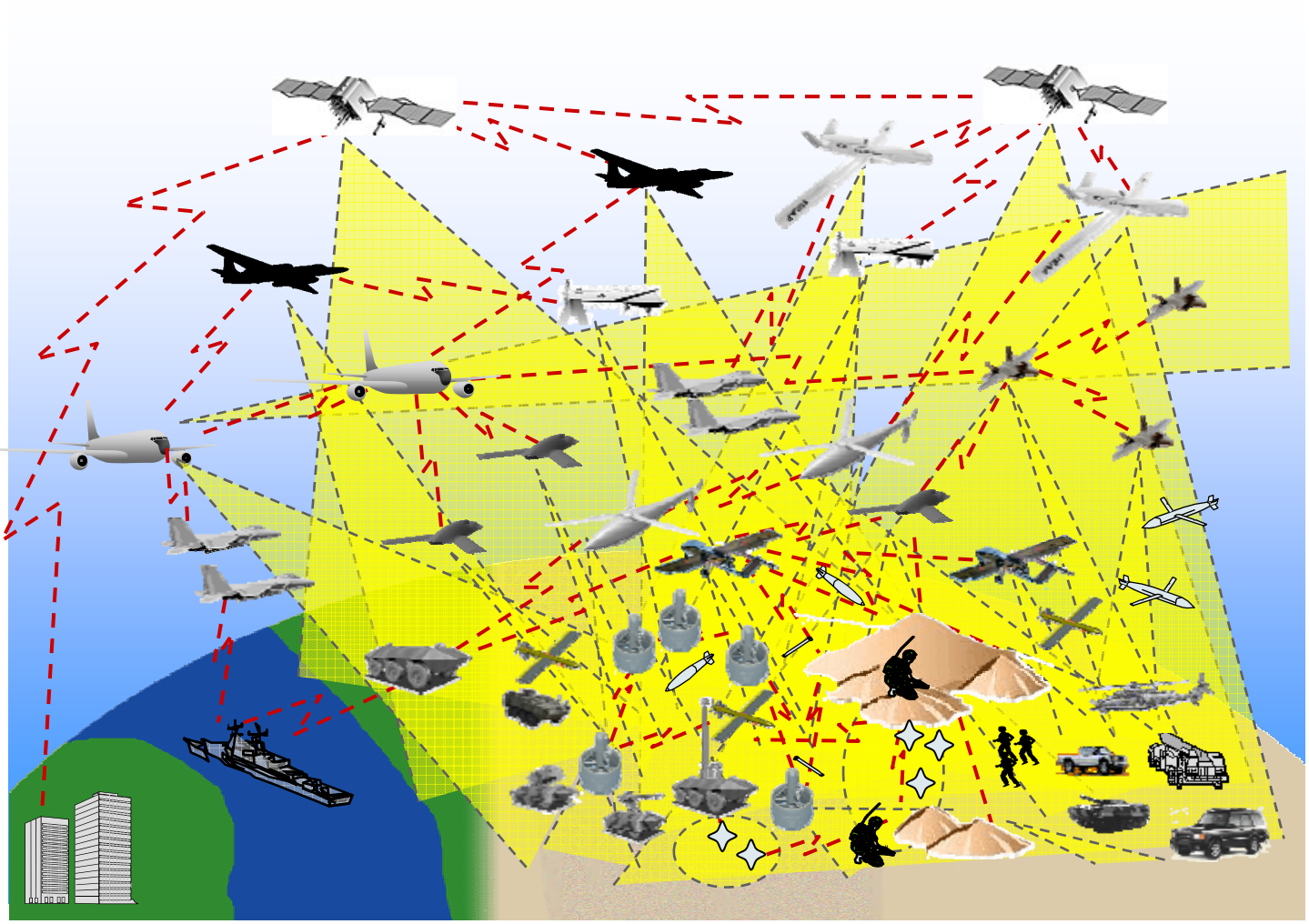
- Agile Computing
  - Agile Computing Overview
  - Mockets Communications Library
  - Group Manager Discovery service
  - FlexFeed Publish-Subscribe System
  - AgServe Service-oriented Architecture
  - DisService Information Dissemination System
- Process Integrated Mechanisms
- Federated Information Spaces
- Work in Progress / Future Ideas
  - DAIMS – Disaster Area Information Management System
  - Information Oriented Networking
  - Network Science

Agile Computing

Agile Computing Definition

- Both a Metaphor and an Approach to Distributed Information Systems
- Design Systems to be
  - Opportunistic in discovering, manipulating, and exploiting computing and communication resources
  - Quick in reacting to changes in the environment
  - Able to take advantage of the “wiggle room” available in the system

The Motivation…(Or Turn Off :-)



Courtesy of Dave Gunning (DARPA)

Domain Characteristics

- Chaotic Environment – Continuously Changing
- Low-bandwidth, variable-latency, unreliable communications (ad-hoc, wireless)
- Unexpected/Unplanned User/Mission Requirements
- Wide range of devices (different architectures, capabilities, power, size)
- Endpoints (clients, sensors) are weak
- Attrition of resources
- Multiple “parallel” activities overlapping in space and time
- Multiple stakeholders (multiple administrative domains)
  - Coalitions, Different Services
- Constraints on resources, usage, and sharing

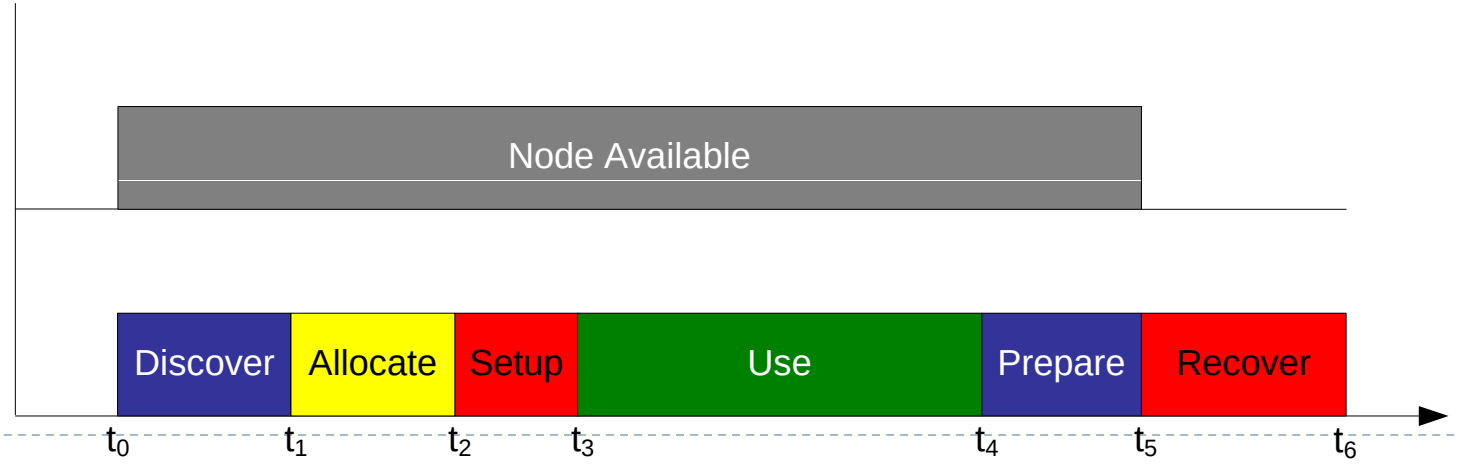
Measuring Agility

- Agility may be measured using
  - How quickly a new resource can be utilized
  - How long a resource needs to be available to be effectively utilized
  - How quickly can a resource be released
- Measures can be used to characterize the *degree of agility* of a system

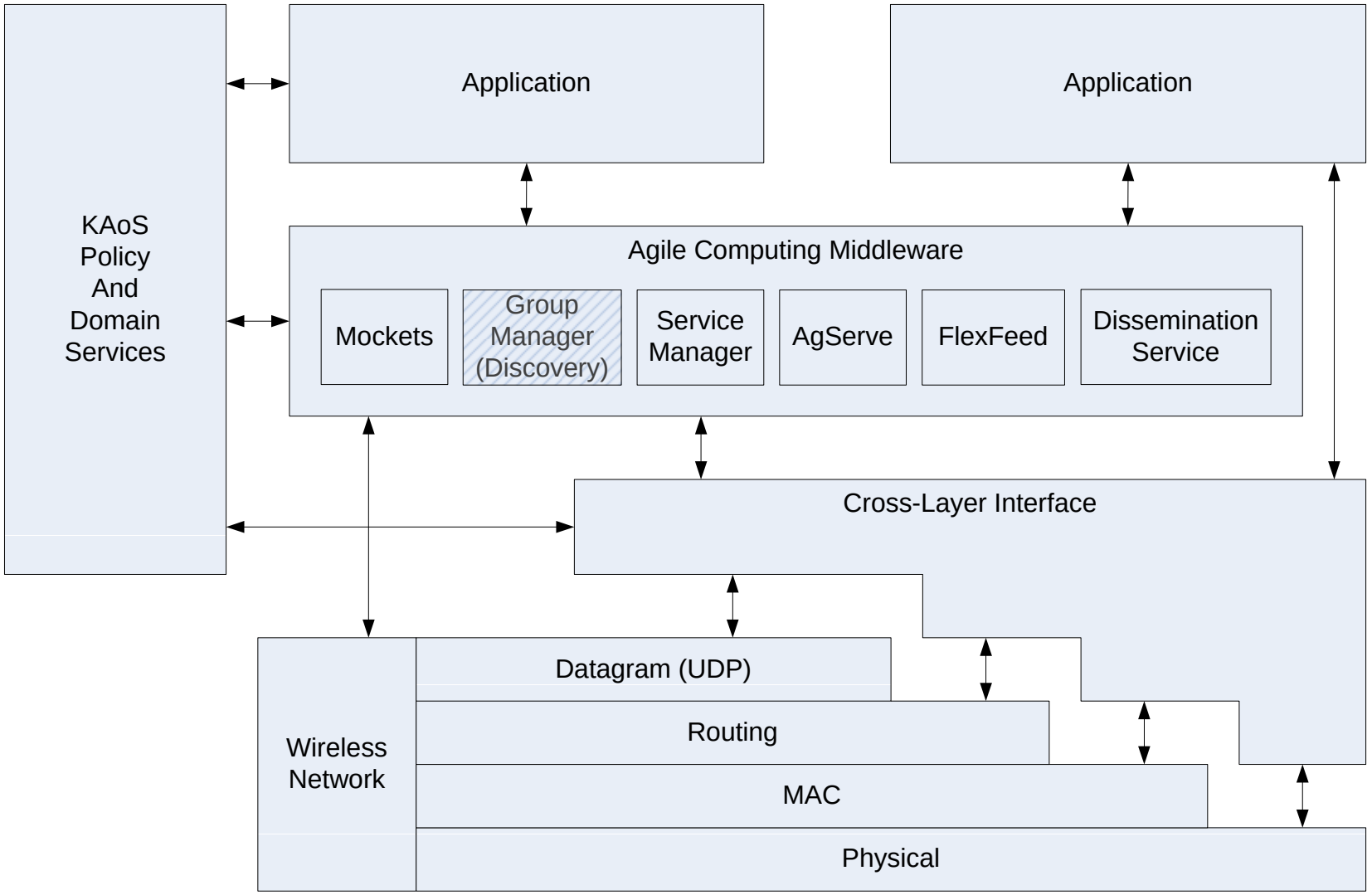
Measuring Agility

Diagram Below Shows Resource Utilization Phases

- t_0-t_4 = Time During Which New Resource is Available
- t_0-t_1 = Time to Discover Resource
- t_1-t_2 = Time to Allocate Resource
- t_2-t_3 = Time to Setup Resource
- t_3-t_4 = Actual Productive Resource Usage Time
- t_4-t_5 = Time to Prepare for Resource Unavailability
- t_5-t_6 = Time to Recover from Resource Unavailability



Components and Context

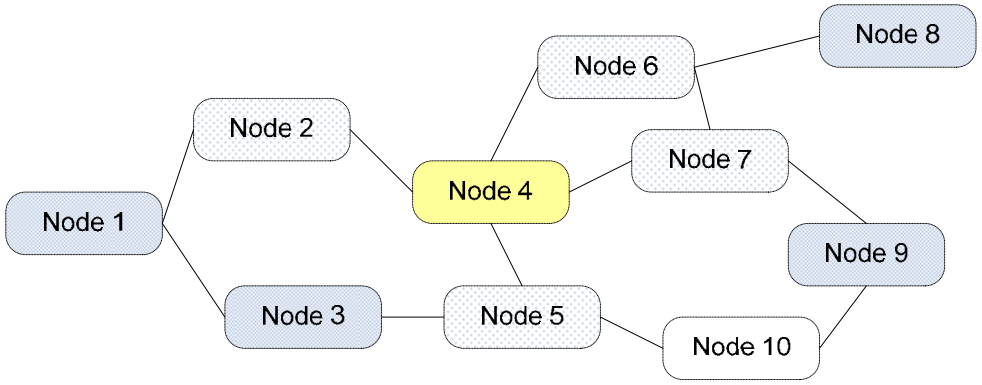


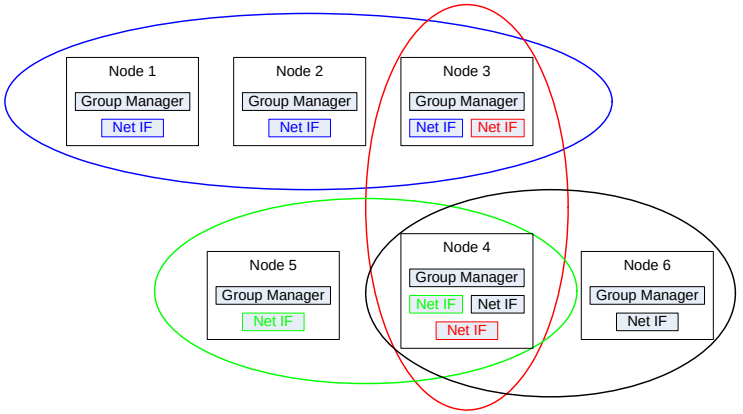
Major Components

- Agile Computing Infrastructure
  - Discovery, Grouping, Tasking, Coordination, Manipulation, Migration, Resource Control
- Mockets
  - Data Agnostic Communications Library, Bandwidth Enforcement, Migration Support
- AgServe
  - Agile Service-oriented Architecture, Dynamic Service deployment, invocation, and migration
- FlexFeed
  - Publish/Subscribe System, Hierarchical Data Distribution, Dynamic Data Transformation, Policy Enforcement
- DisService
  - Peer-to-peer Information Dissemination Service
- Cross-cutting component – policy infrastructure

Group Manager

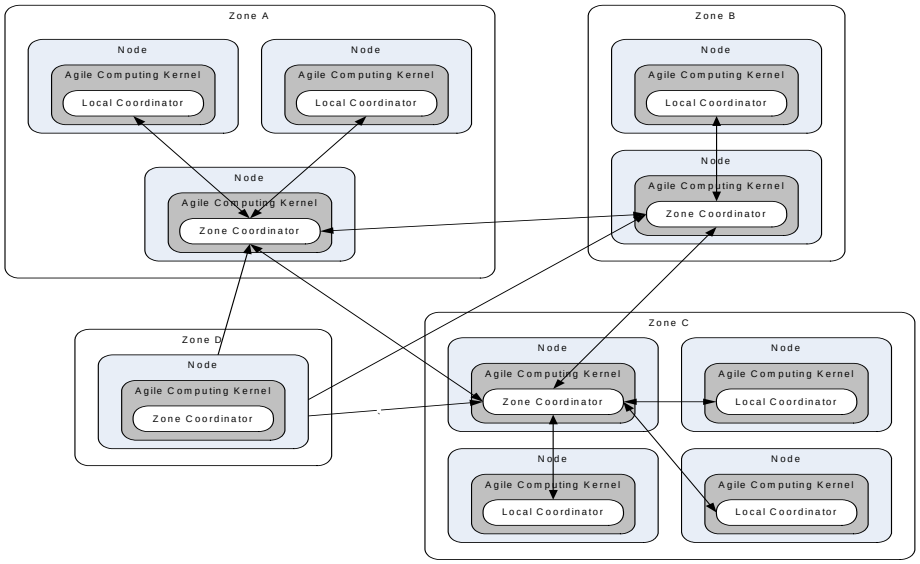
- Group Manager Supports Node and Resource Discovery
- Designed to Fit Well in MANET Environments
- Supports Managed and Peer Groups, with Optional Security Mechanisms
- Flexible in Allowing Nodes to Advertise Weakly or Strongly
- Flexible in Allowing Arbitrary Propagation of Information
- Supports combination of both proactive and reactive modes
  - Proactive mode – push information out to specified radius
  - Reactive mode – search information within specified radius
- Supports Peer-to-Peer Search Using a Modified Gnutella-style Algorithm

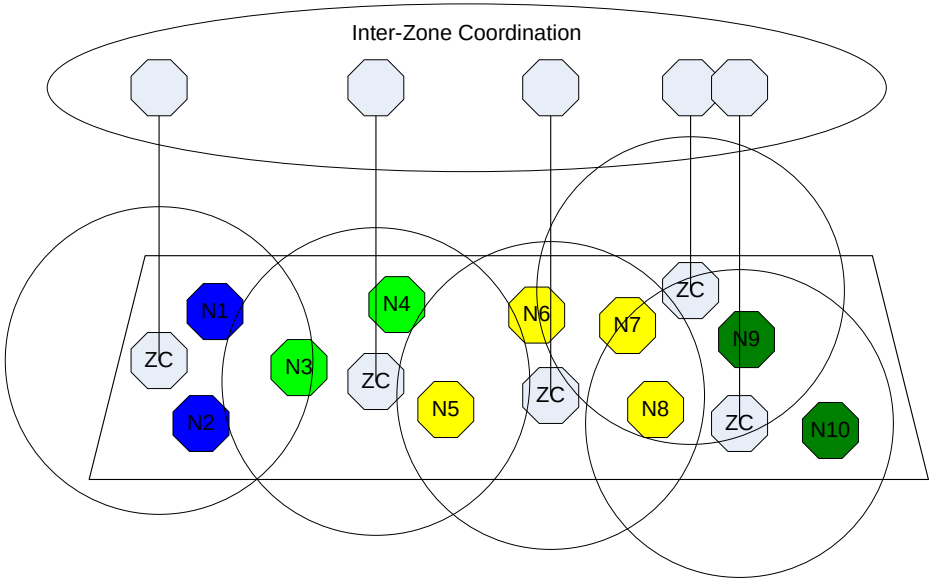




Group Manager

- Integrated with MANETs using the cross-layer interface
  - Reduce information propagation overhead and improve reactivity
- Provides the foundation for hierarchical organization of nodes
  - Example – Zone-based coordination



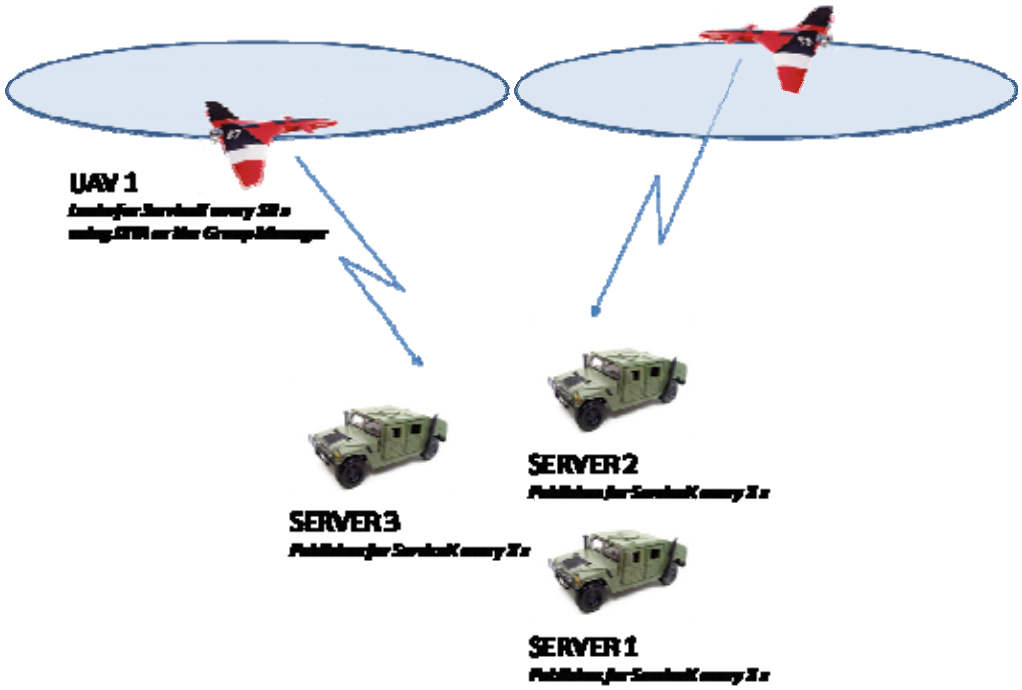


Service Discovery Results

- Compare Service Manager / Group Manager with JXTA
  - Measure bandwidth utilized for service discovery operation
  - Use a MANET configuration with intermittent connectivity
    - Target environment for Agile Computing, SOSCOE
    - JXTA uses a Distributed Hash Table – not well suited to networks with a high churn rate
    - But – JXTA is still proposed for discovery in such environments

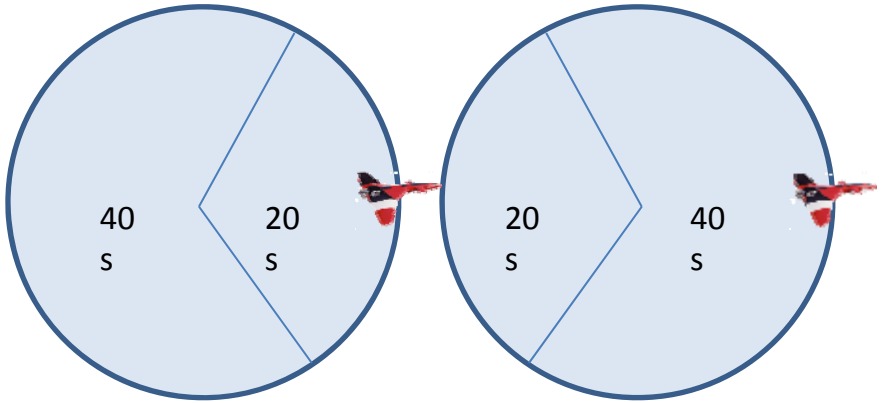
Scenario

- Network with Three - Ten nodes
  - Two to Four UAVs
  - One to Six Ground Nodes



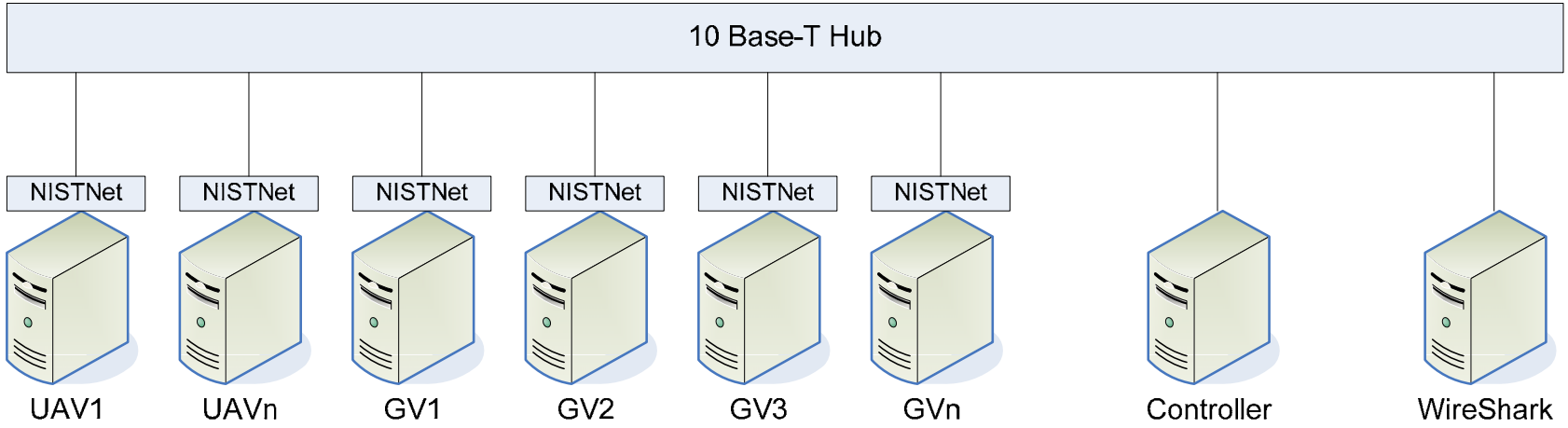
Network Connectivity

- No Connectivity Between UAVs
- Full Connectivity Between Ground Vehicles
- Each UAV goes through the following phases with the Ground Vehicles
  - No connectivity for 40 seconds
  - Some connectivity for 7 seconds (30% packet drop)
  - Good connectivity for 6 seconds (10% packet drop)
  - Some connectivity for 7 seconds (30% packet drop)
- UAVs are in opposite “phase” with each other



Testbed Configuration

- Consists of 12 machines interconnected via a hub
- Uses NISTNet to control network traffic
- Uses a hub and WireShark to observe traffic



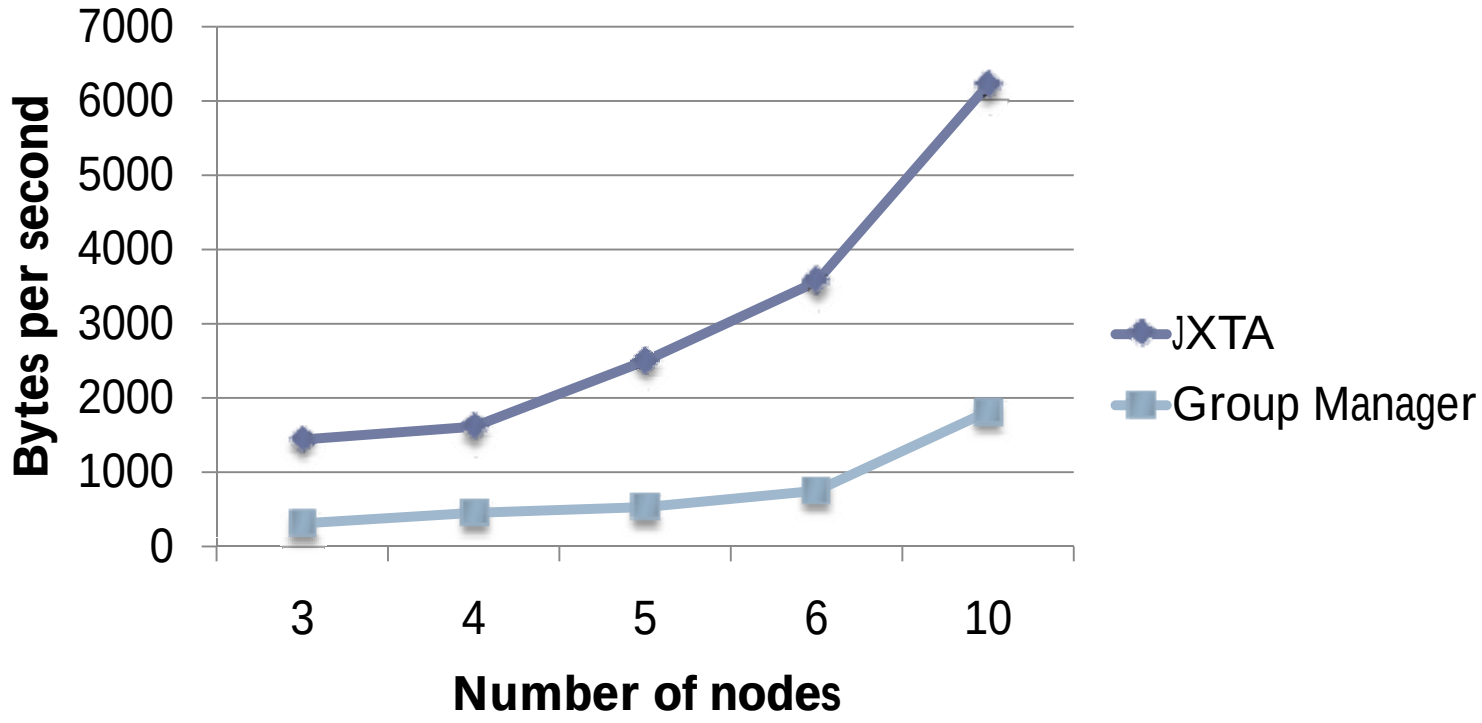
Pentium III Class (or better) – Running Fedora Core 4

Ten Node Result

System	Destination										
Sender	Muticast	GV1	GV2	GV3	GV4	GV5	GV6	UAV1	UAV2	UAV3	UAV4
JXTA-E											
GV1	187.75	--	--	--	--	--	--	48.53	88.98	94.45	33.63
GV2	188.03	--	--	--	--	--	--	22.07	87.74	97.05	181.14
GV3	187.08	--	--	--	--	--	--	216.88	96	63.81	181
GV4	187.67	--	--	--	--	--	--	171.36	--	169.73	198.83
GV5	186.86	--	--	--	--	--	--	22.07	85.03	240.01	183.46
GV6	187.74	--	--	--	--	--	--	22.07	91.75	159.37	199.04
UAV1	432.18	--	--	56.92	44.13	--	--	--	--	--	--
UAV2	432.86	--	--	20.97	--	--	1.7	--	--	--	--
UAV3	432.42	27.59	23.94	40.31	7.98	7.67	7.24	--	--	--	--
UAV4	432.66	--	45.94	134.74	122.01	52.75	9.65	--	--	--	--
Total	**6213**										
GM											
GV1	--	--	--	--	--	--	--	72.77	72.76	72.76	72.8
GV2	--	--	--	--	--	--	--	60.64	60.64	62.8	60.65
GV3	--	--	--	--	--	--	--	52.77	52.75	52.76	53.38
GV4	--	--	--	--	--	--	--	47.1	46.9	46.9	46.1
GV5	--	--	--	--	--	--	--	95.38	104.61	103.43	94.5
GV6	--	--	--	--	--	--	--	37.7	37.68	37.68	37.7
UAV1	83.45	--	--	--	--	--	--	--	--	--	--
UAV2	83.61	--	--	--	--	--	--	--	--	--	--
UAV3	83.41	--	--	--	--	--	--	--	--	--	--
UAV4	83.65	--	--	--	--	--	--	--	--	--	--
Total	**1817**										

Bandwidth Utilization Trend

- Group Manager / Service Manager Appears to Scale Better
- Still need to try with more (>20) nodes



Mockets Communications Library

- Overview
  - Application-level Communications library
  - Replaces TCP and UDP
  - Available for C++, Java, and C#
  - Supports Stream and Message abstractions, with multiple types of service
  - Integrates with KAoS for Bandwidth Control
- Benefits
  - Better performance than TCP
  - Enhanced API provides new capabilities
    - Message Tagging
    - Replacement
    - Prioritization
  - Detailed statistics and feedback support application adaptation
  - Easy to integrate into existing applications that use sockets
  - Flexible – allows easy modifications and customizations
  - Bandwidth control
  - Endpoint migration and terminal mobility

Experimental Evaluation

- Compared throughput between mockets and TCP sockets
  - Measured time to transmit 2 MB chunk
- Network configurations
  - 802.11b Ad-hoc mode
  - BBN Ad-hoc radio nodes (Radio A)
    - One, two, three, and four hop configurations
  - LinkSys Ad-hoc radio nodes (Radio B)

Experimental Evaluation

- Throughput (bytes/ms) comparison between mockets and sockets

	Mockets	Sockets	Improvement
Configuration	Throughput (bytes/ms)		
802.11b Ad-Hoc	503.02	410.02	22.68%
Radio A 1 Hop	856.53	847.35	1.08%
Radio A 2 Hops	345.30	292.64	17.99%
Radio A 3 Hops	164.98	147.65	11.73%
Radio A 4 Hops	97.99	80.70	21.43%
Radio B 1 Hop	423.49	370.58	14.28%
Radio B 2 Hops	394.17	318.33	23.83%

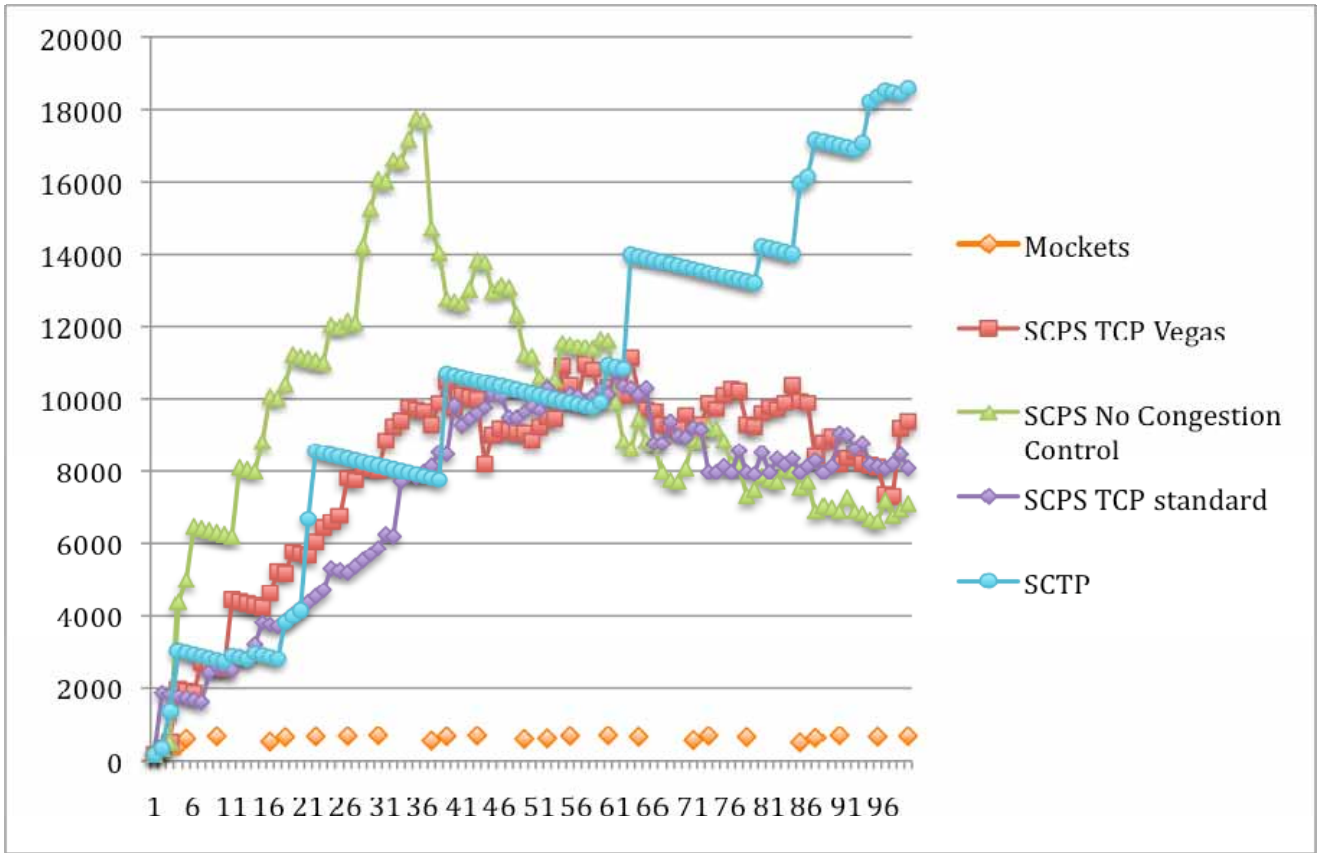
Experimental Evaluation

- Flight Test – J-STARS Communicating with a Ground Station over PRC-117 Radio

	Mockets	Sockets	Improvement
	Throughput (bytes/sec)		
56 Kbps	3292.43	431.63	762.80%
Emulation			

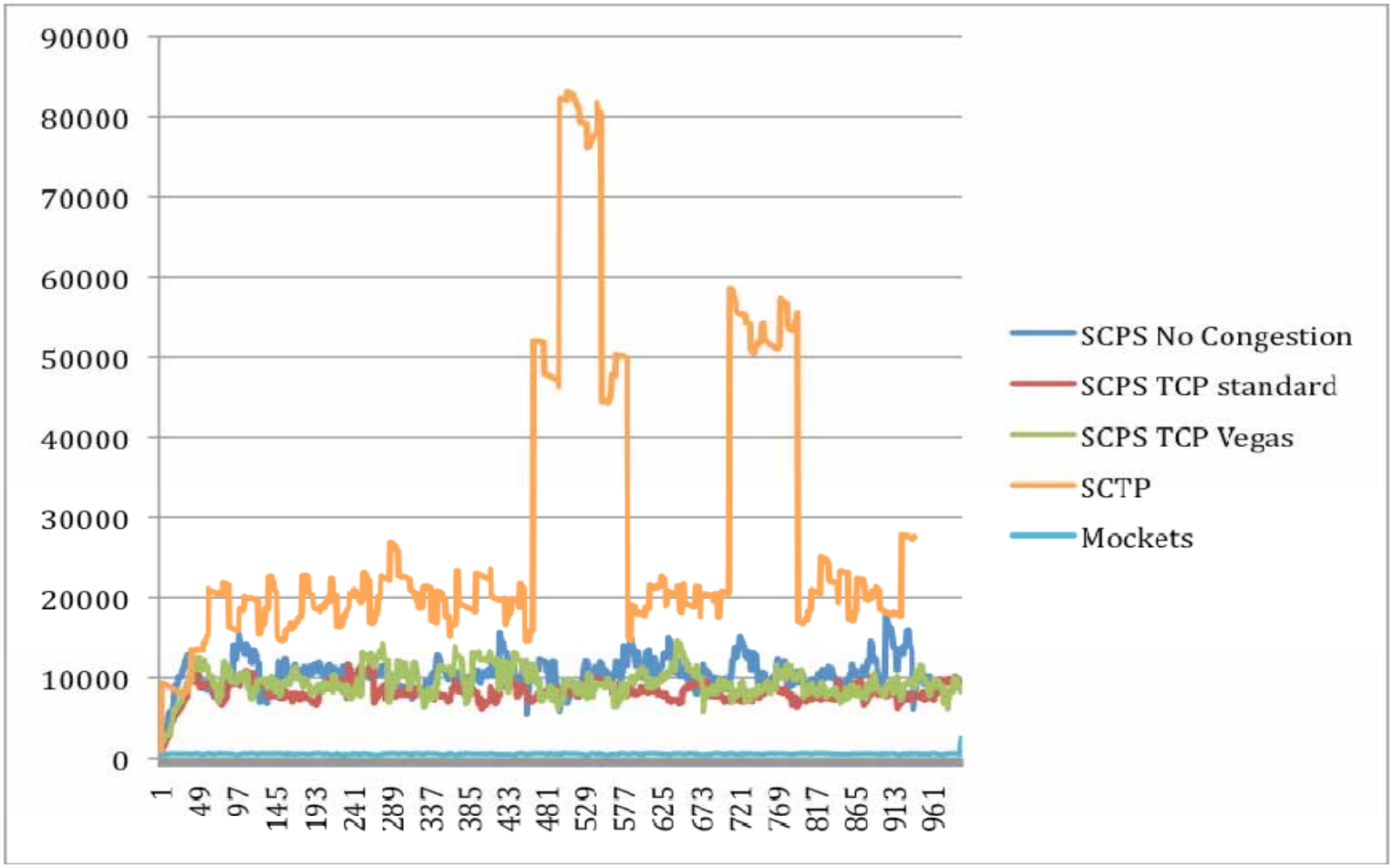
Experimental Evaluation

- Evaluated Mockets for Transport in JBI
  - Measured Timeliness of Arrival of Data



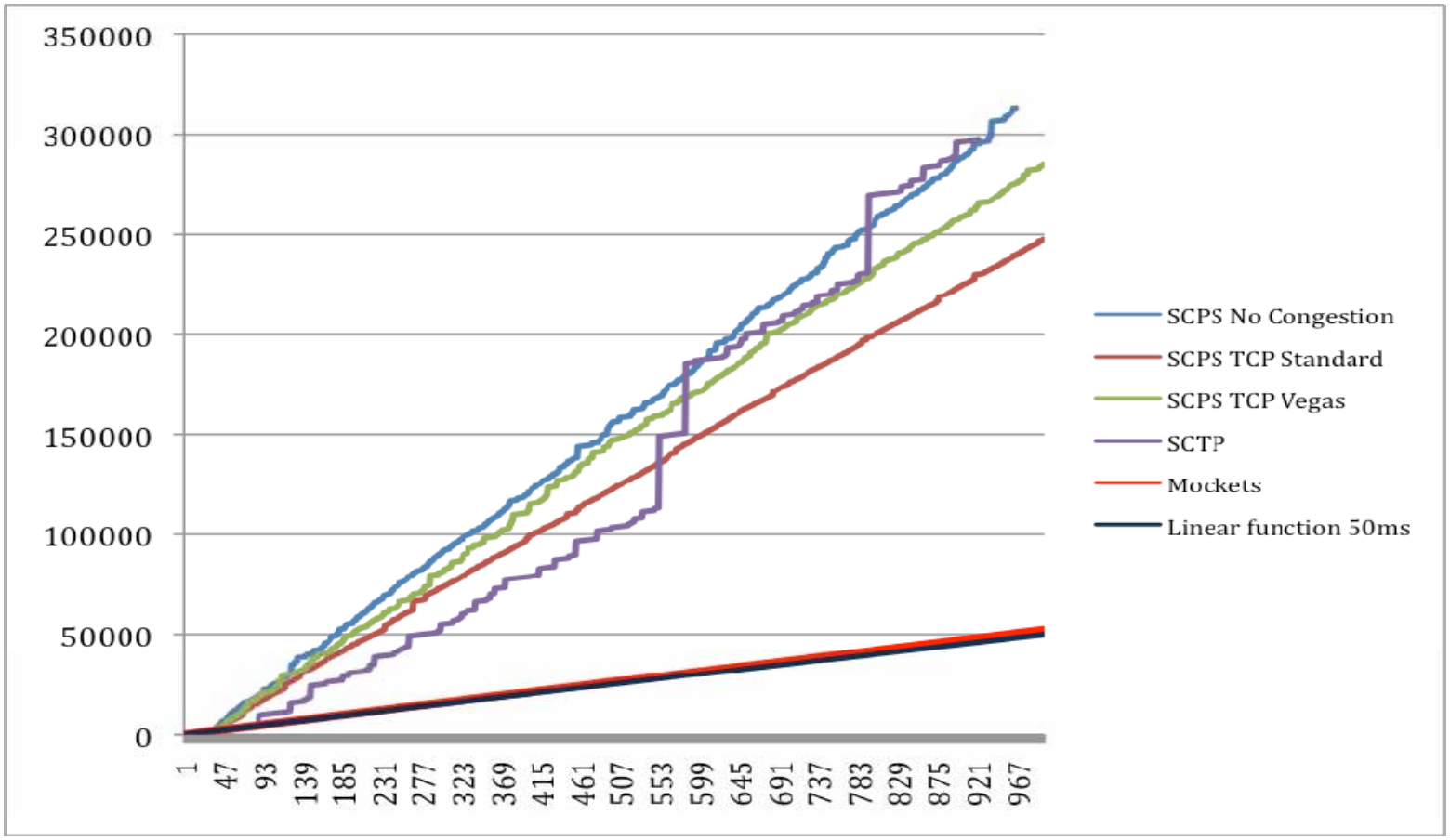
Experimental Evaluation

- Results for 1000 messages

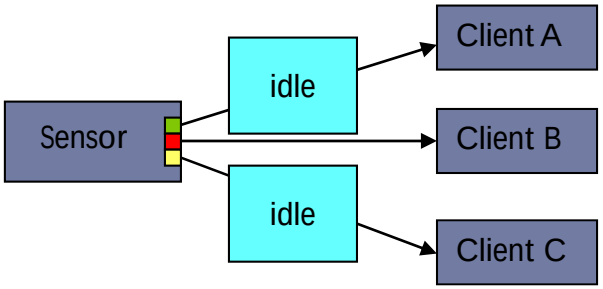


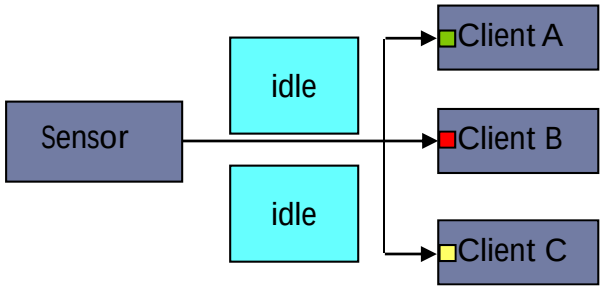
Experimental Evaluation

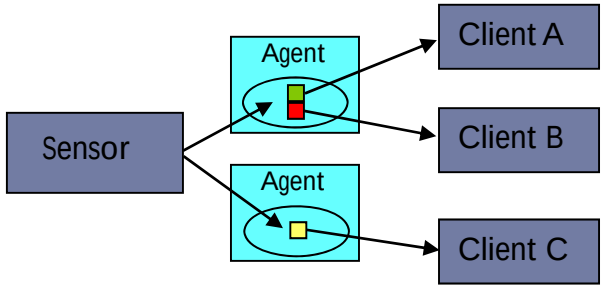
- Results for 1000 messages – Arrival Latency



FlexFeed: Efficient Data Distribution



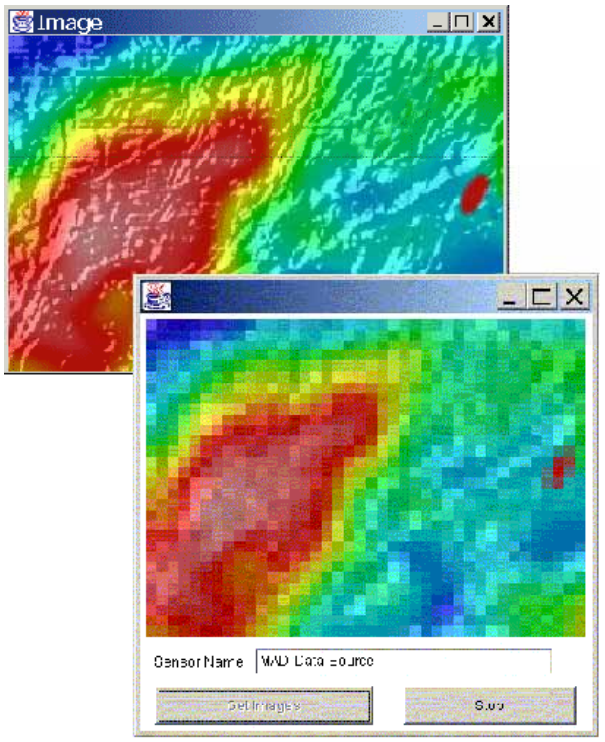




- Distribute Processing along the data path
- Optimize Bandwidth
- Construct Deterministic and Adaptive Data Distribution Paths

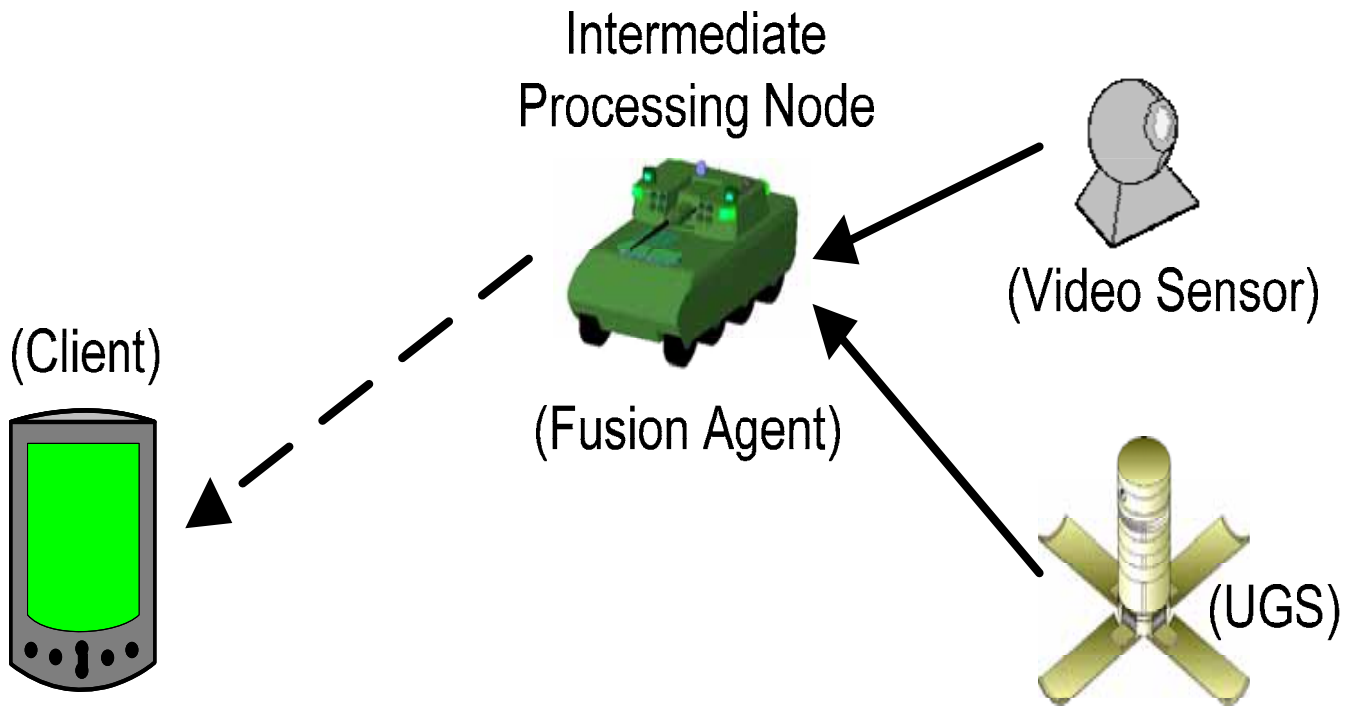
FlexFeed: Policy Enforcement

- Dynamically Transform Data Enroute from Producers to Consumers
  - Transformation is Embedded in the Network Along the Path
- Example with Video Data
- Policies Can Control
  - Resolution
  - Frame-rate
  - Real-time
  - Redaction / Scrubbing



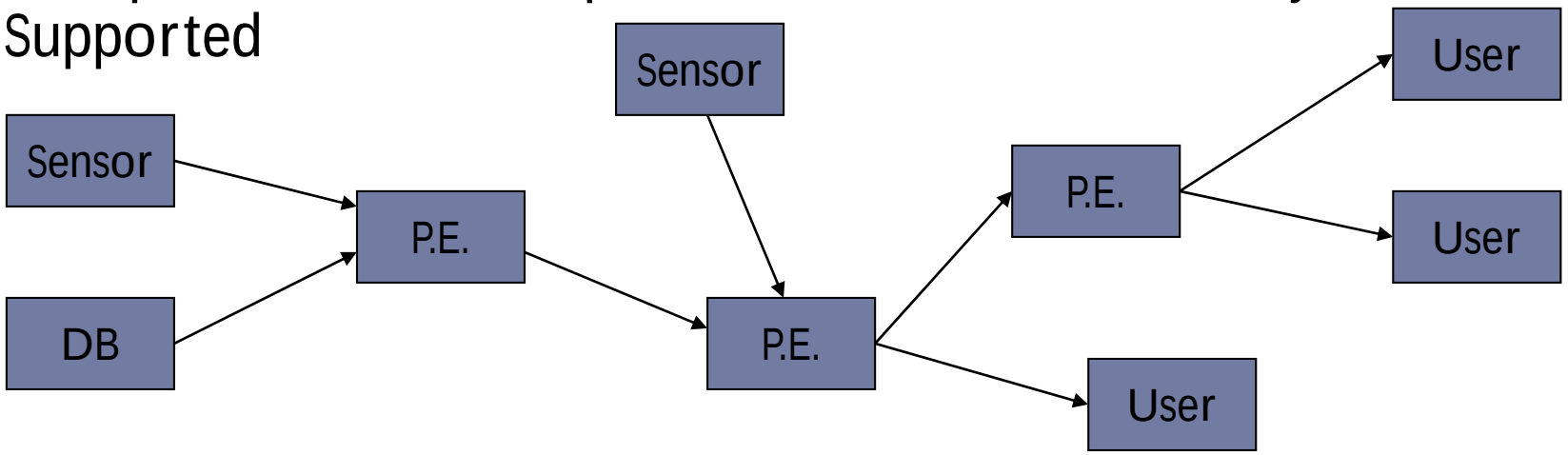
FlexFeed: Data Fusion

- FlexFeed Opportunistically Finds Available Resources for the Fusion Component



FlexFeed

- FlexFeed Generates and Manages Data Flow Graphs
  - Specifies data sources, processing elements, and data sinks
  - Also specifies bandwidth and processing requirements
- Data Flow Graphs Are Mapped Dynamically to the Environment
  - Processing elements are allocated to intermediate nodes based on communications and resource availability
- Multiple Data Flow Graphs Need to be Concurrently Supported



AgServe

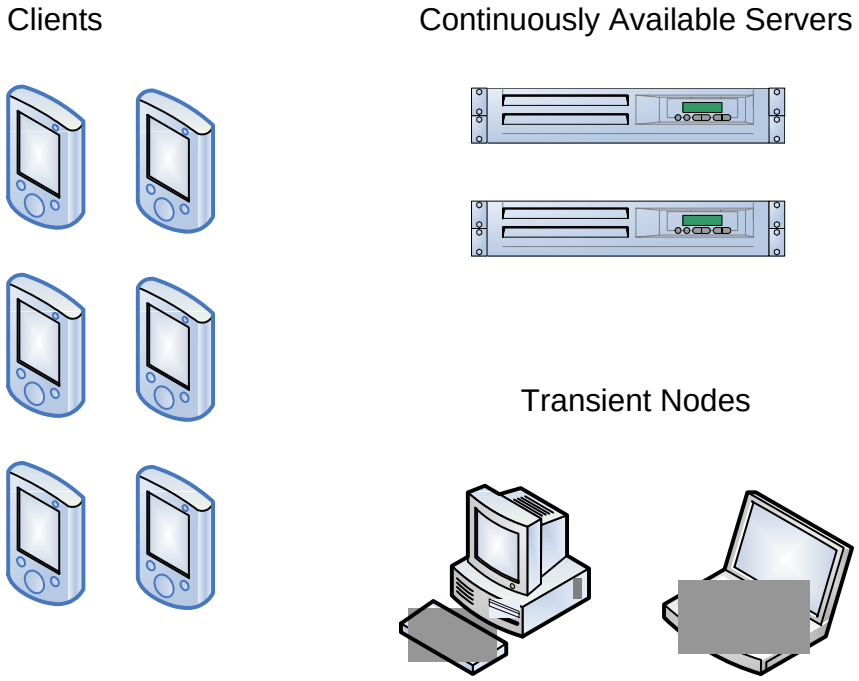
- Service-oriented architecture for tactical environments
- Support dynamic services
  - Definition, instantiation, invocation, and relocation
- Monitor service resource utilization and service invocation patterns
  - Learn resource profiles and invocation patterns
  - Resources include CPU, memory, storage, comms
- Dynamically manage service instances and invocation mappings

Service Migration Results

- Reminder - Agility may be measured using
  - How quickly a new resource can be utilized
  - How long a resource needs to be available to be effectively utilized
  - How quickly can a resource be released
- Measures can be used to characterize the *degree of agility* of a system

Service Migration Experiment

- Experimentally determine first two agility measures
  - Time to take advantage of a new resource
  - Minimum time necessary to gain benefit



Service Migration Results

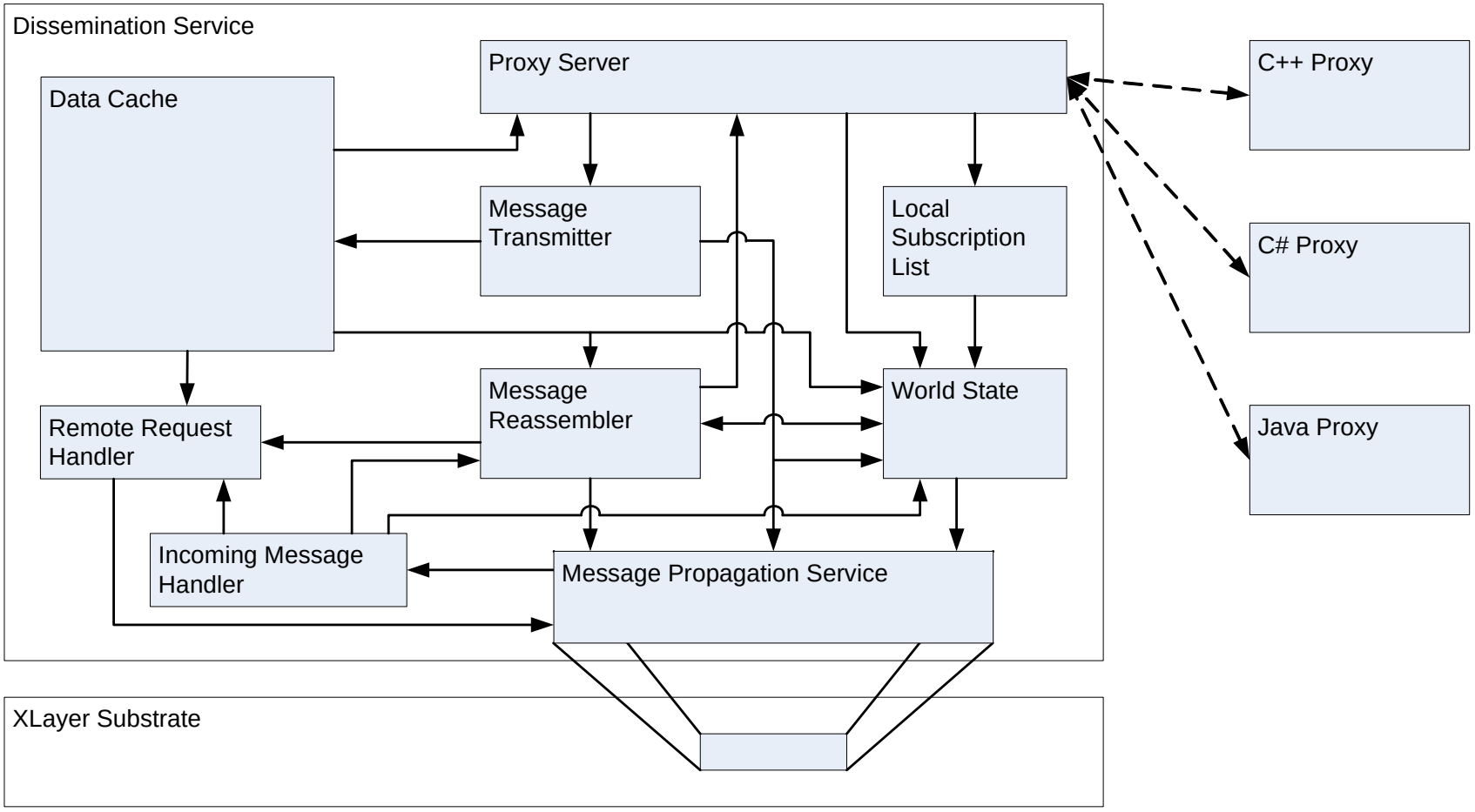
- Service migration starts providing benefit (reduction in overall execution time) when transient node is available for > 8 seconds
- Need to measure with additional services

Transient Node	Service #1	Service #2	Overall
Time (ms)	(Execution Time in ms)		
0	98362	98570	98570
5000	99421	100171	100171
10000	96683	97337	97337
15000	92019	93443	93443
20000	88990	89203	89203
30000	79324	81220	81220
40000	75322	78208	78208
50000	64870	70686	70686
60000	58614	66706	66706
70000	58218	64826	64826
Baseline time for service execution - 54760 ms			

DisService – Tactical Data Dissemination

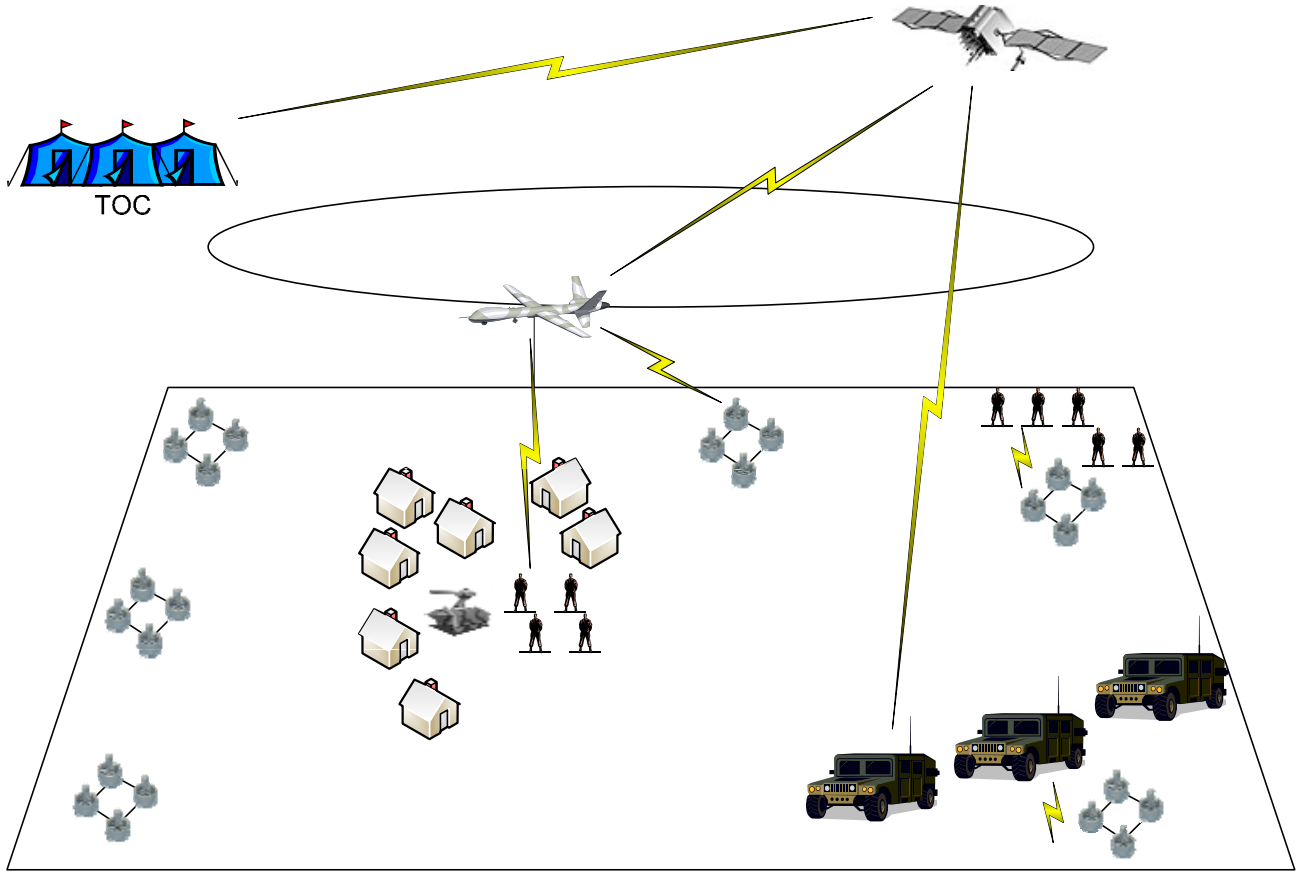
- Combat Situations Require Timely and Efficient Dissemination of Data
  - Dissemination should be peer-to-peer
  - Well suited for mobile ad-hoc networks
- Three broad categories
  - Situation Awareness (SA) Data – widely disseminated
  - Directed Data – usually sent to a small subset of nodes
  - On-demand Data – only sent upon request
- Dissemination Capability Should Support all Three Modes

DisService Architecture



Scenario – Sensor Data Harvesting

- Collecting and Disseminating Data from Sensor Networks



DisService Capabilities

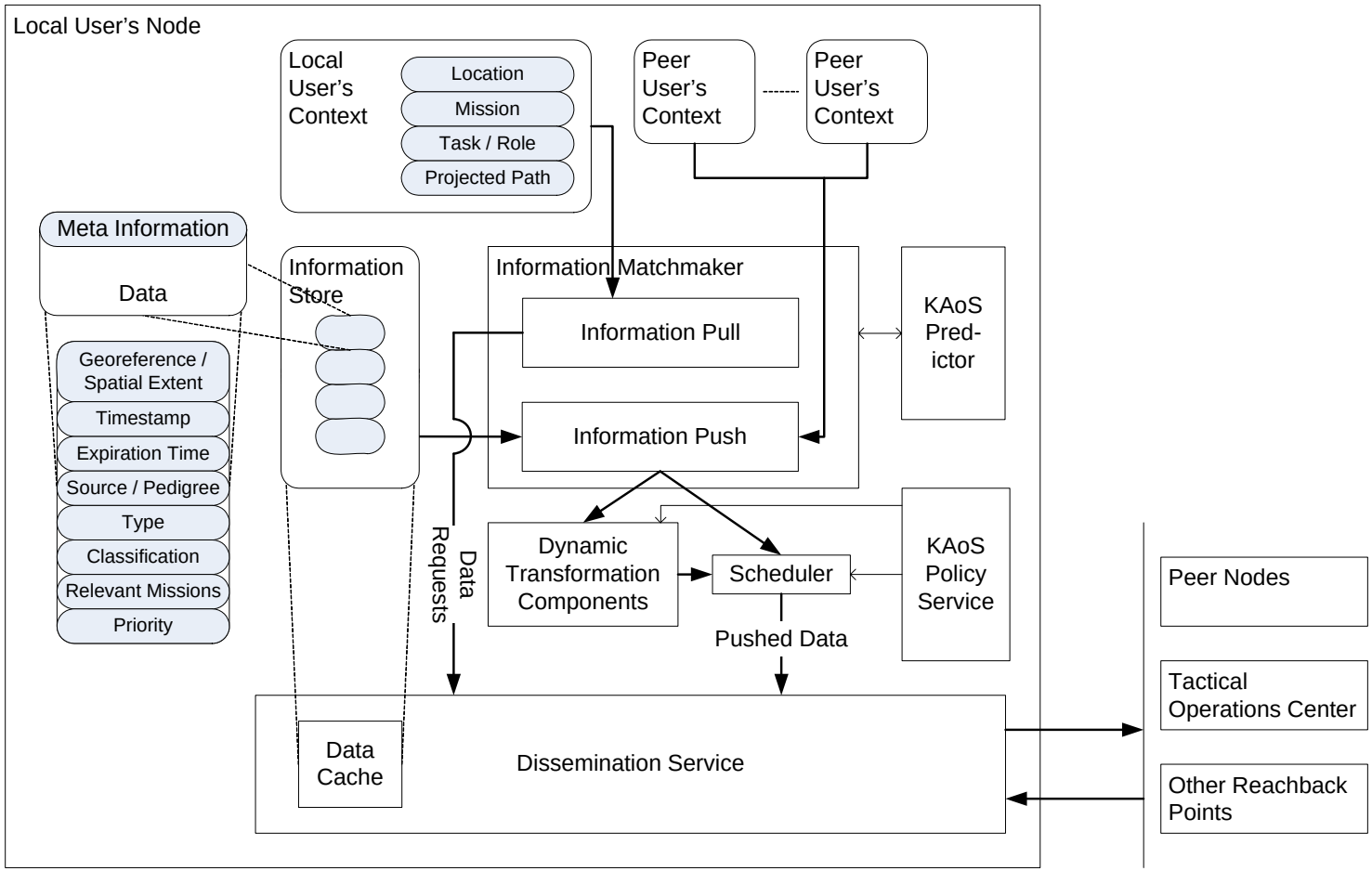
- Support dissemination of data
  - Small or large in size
  - With reliable or unreliable delivery
  - With sequenced or unsequenced delivery
  - Using store-and-forward, to support partitioned networks
  - With prioritization, to handle insufficient bandwidth (transmission) and capacity (storage)
  - Scoped by hierarchical groups, to support organization
  - With tagging, to differentiate between types of messages
  - With dynamic adaptation, to support multiple dissemination patterns
    - One-to-one, One-to-many, many-to-many, many-to-one, many-to-few, few-to-many, few-to-few, one-to-few, few-to-one
  - With dynamic resource discovery and exploitation
    - Comms for replication, Storage for caching
  - With application control over management and propagation of data

Proactive Dissemination

- Increase Information Availability and Reduce Latency by Prestaging Information
- Integrate
  - Information Anticipation
  - Proactive Pushing of Information
  - Information Prioritization
  - Efficient Dissemination
  - Information Adaptation
  - Exploitation of Peer Resources

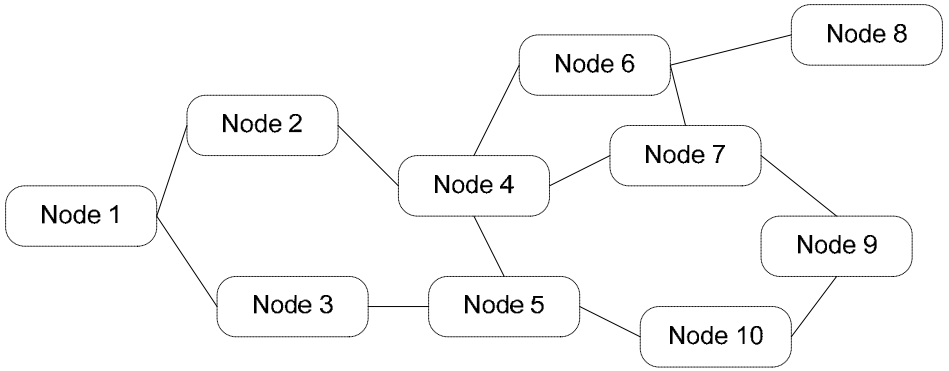
Proactive Dissemination

- Architecture



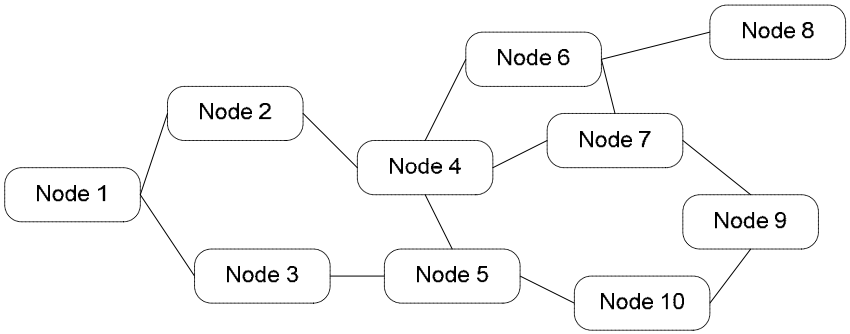
Reliable Reception Instead of Reliable Transmission

- TCP and Other Reliable Transport Protocols Depend on Sender to Ensure Reliability
- Not Appropriate for DisService
  - One Sender, Multiple Receivers
  - Different Requirements for Different Receivers
    - Each receiver independently chooses reliable or unreliable subscriptions
  - Receivers may come and go
  - Multiple nodes may satisfy a receiver's request
- Instead, DisService Relies on Recipients Actively Requesting Missing Messages/Fragments



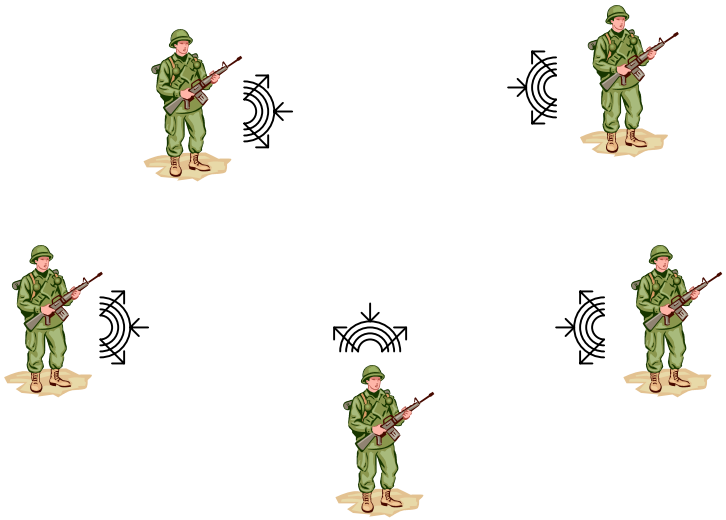
Neighbor Dependent Probabilistic Response Model

- Each Node may have Multiple Peers
- Requests from a Peer may be Received by Multiple Nodes
  - Example: a request for a missing message
- Creates Unnecessary Network Traffic if Multiple Nodes Reply
- Instead, DisService Nodes Responds Probabilistically
  - Each request includes neighbor count
  - Probability of response is inverse of neighbor count



DisService Results (1)

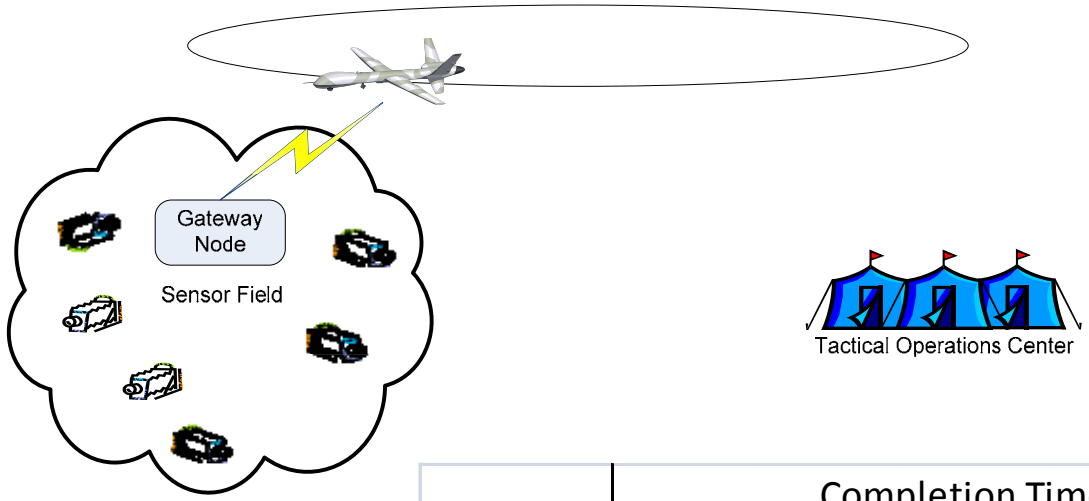
- SA Data Scenario



	(Total Bandwidth)		(Bandwidth Rate)	
Reliability	DS (KB)	TCP (KB)	DS (KB/s)	TCP (KB/s)
100%	6101.68	23073.39	9.71	38.44
90%	10847.51	25956.77	17.16	43.20
80%	13234.57	27945.89	20.73	36.12

DisService Results (2)

- Data Harvesting Scenario



	Completion Time (Seconds)					
	DS	TCP	DS	TCP	DS	TCP
Reliability	(10 Mbps Band.)		(500 Kbps Band.)		(250 Kbps Band.)	
100%	10.65	7.25	29.23	27.87	50.85	47.81
90%	28.44	21.87	46.05	50.03	63.75	73.86
80%	50.78	497.04	51.39	1485.56	81.37	
70%	70.18		77		97.49	
60%	102.87		119.48		119.92	
50%	143.61		170.17		142.5	

Process Integrated Mechanisms

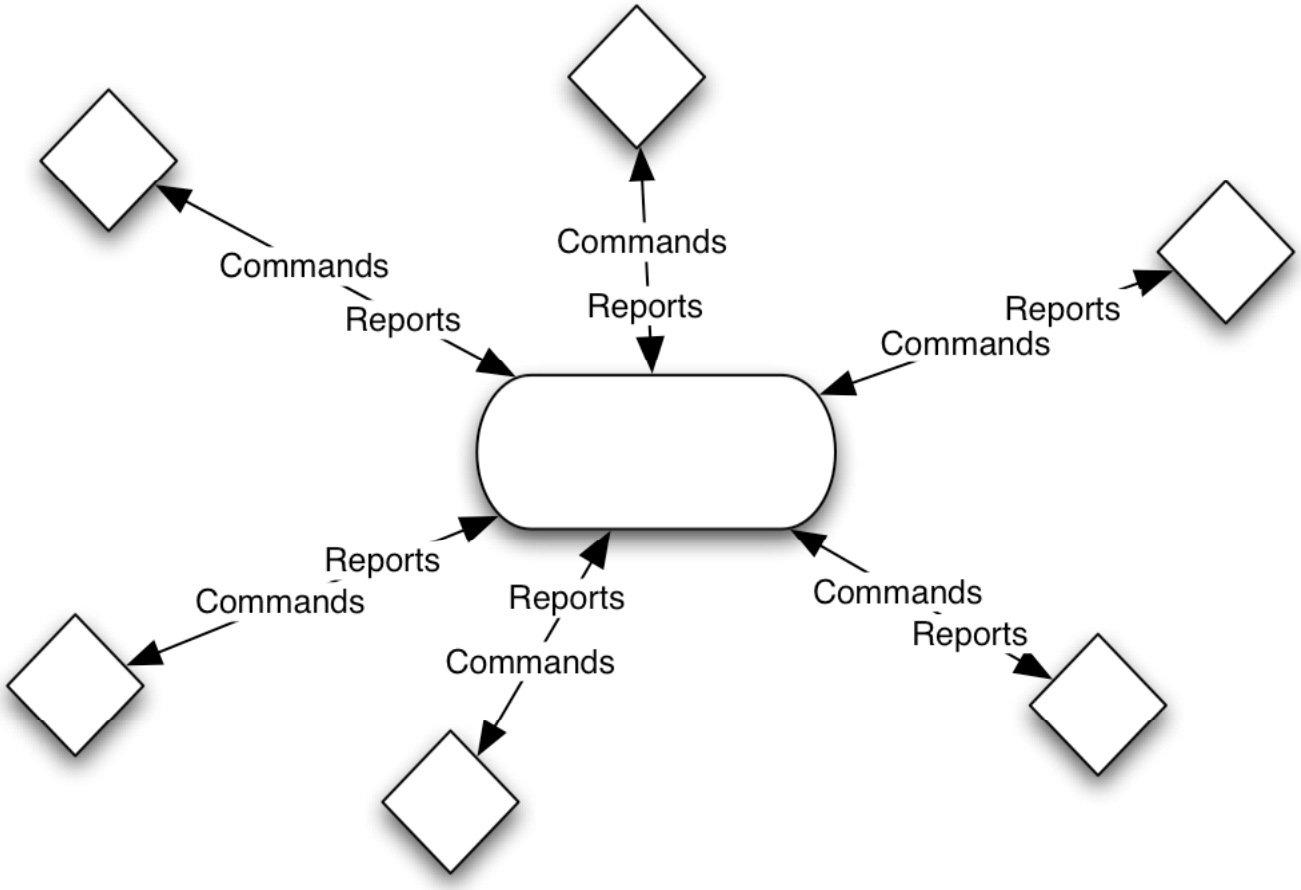
The Problem

- Coordination of robots, UAVs, sensors…
  - Search and rescue
  - Scouting
  - Construction (e.g., space telescope)
  - Combat
  - Scatterbots…distributed mechanisms comprised of non-homogeneous parts.

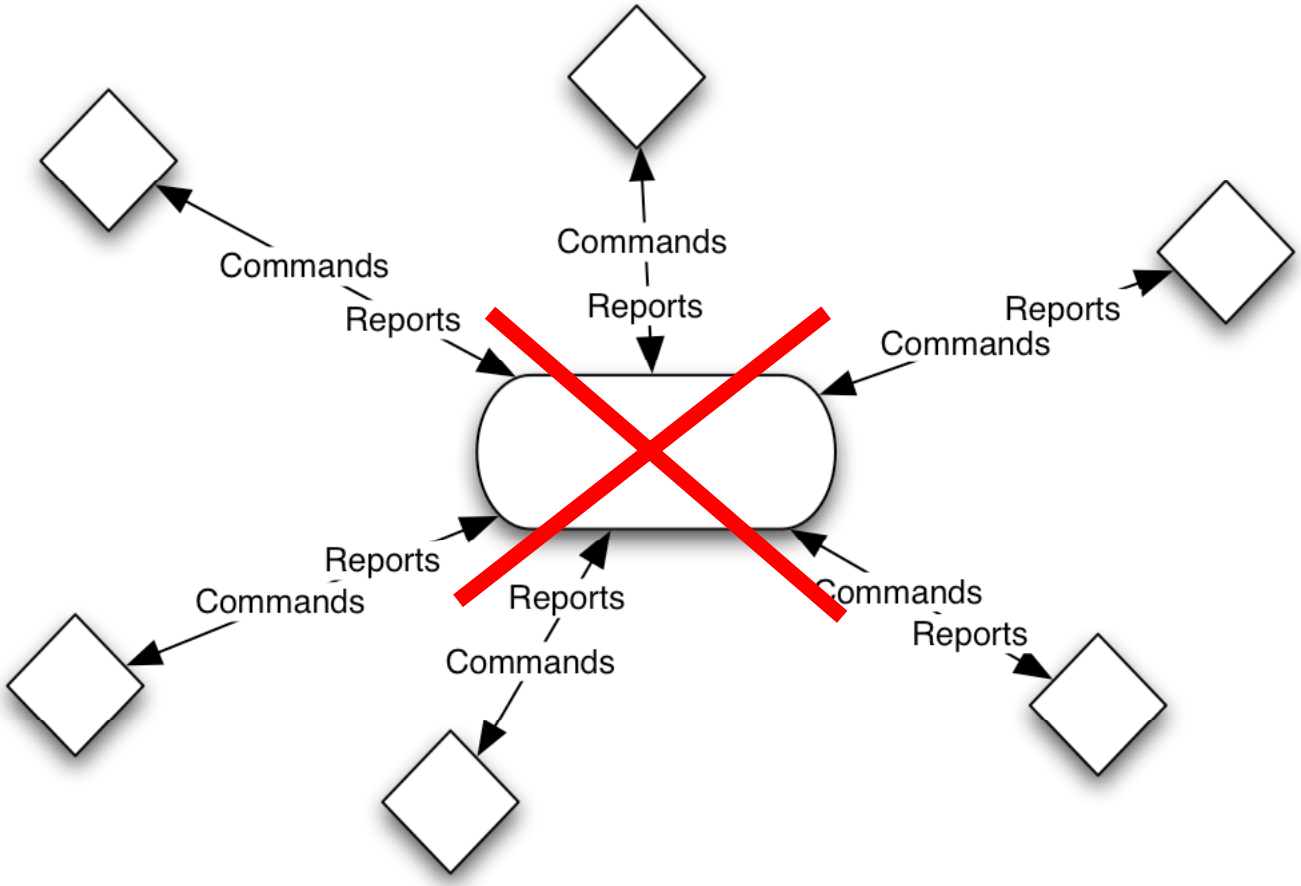
Predominant Approaches

- Centralized Control
  - Maintains a global view from reports
  - Dispatches commands to robots
- Swarms of autonomous robots
  - Group behavior is an “emergent” behavior
  - No global view
  - Hard to control and predict
- Agent-based Distributed Teams
  - Communicating robots “negotiate” to determine group activity
  - No global view

Centralized Control

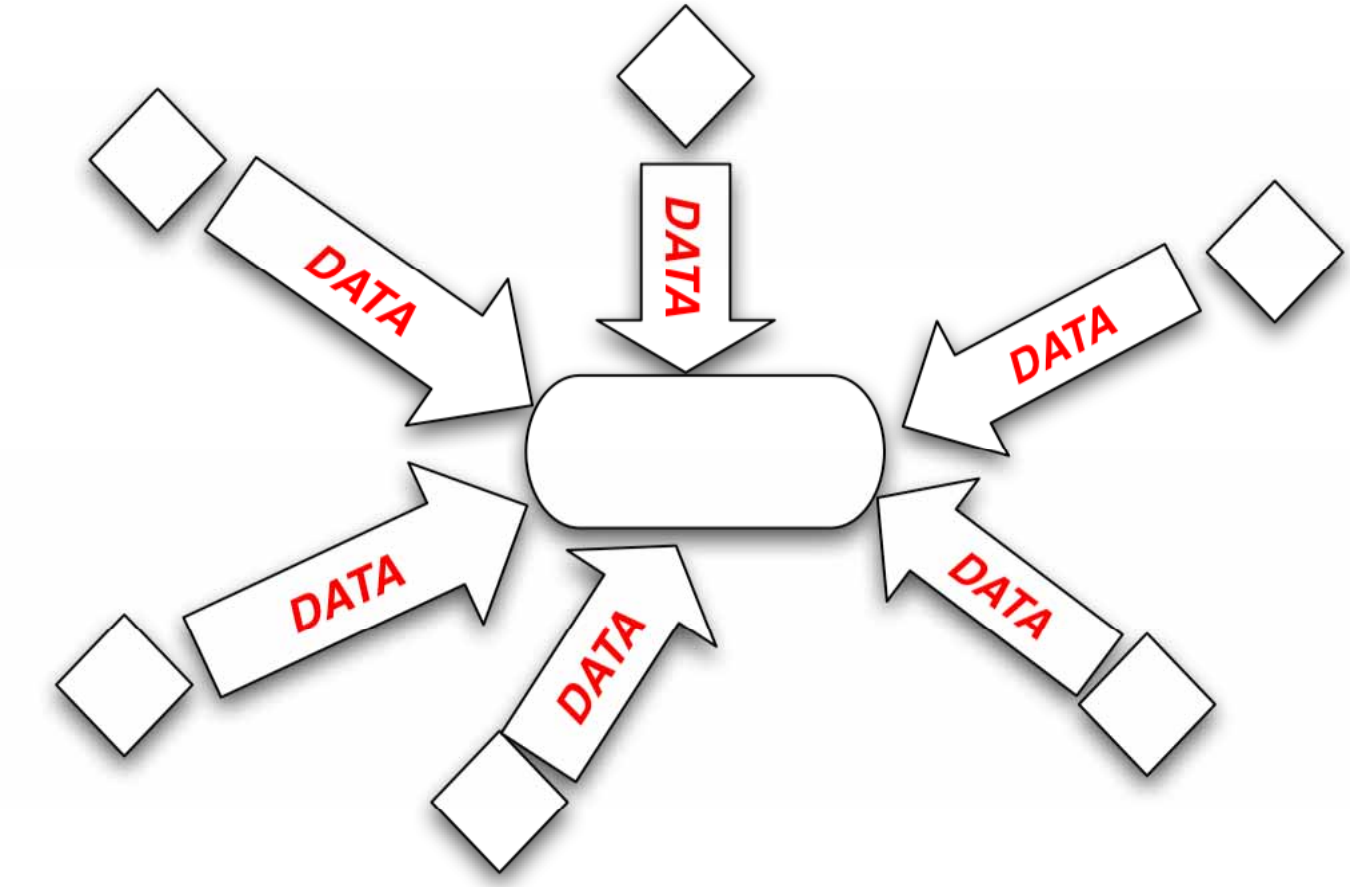


Centralized Control: Problems



Fragile:Whole system disabled if central controller destroyed or inoperational

Centralized Control: Problems



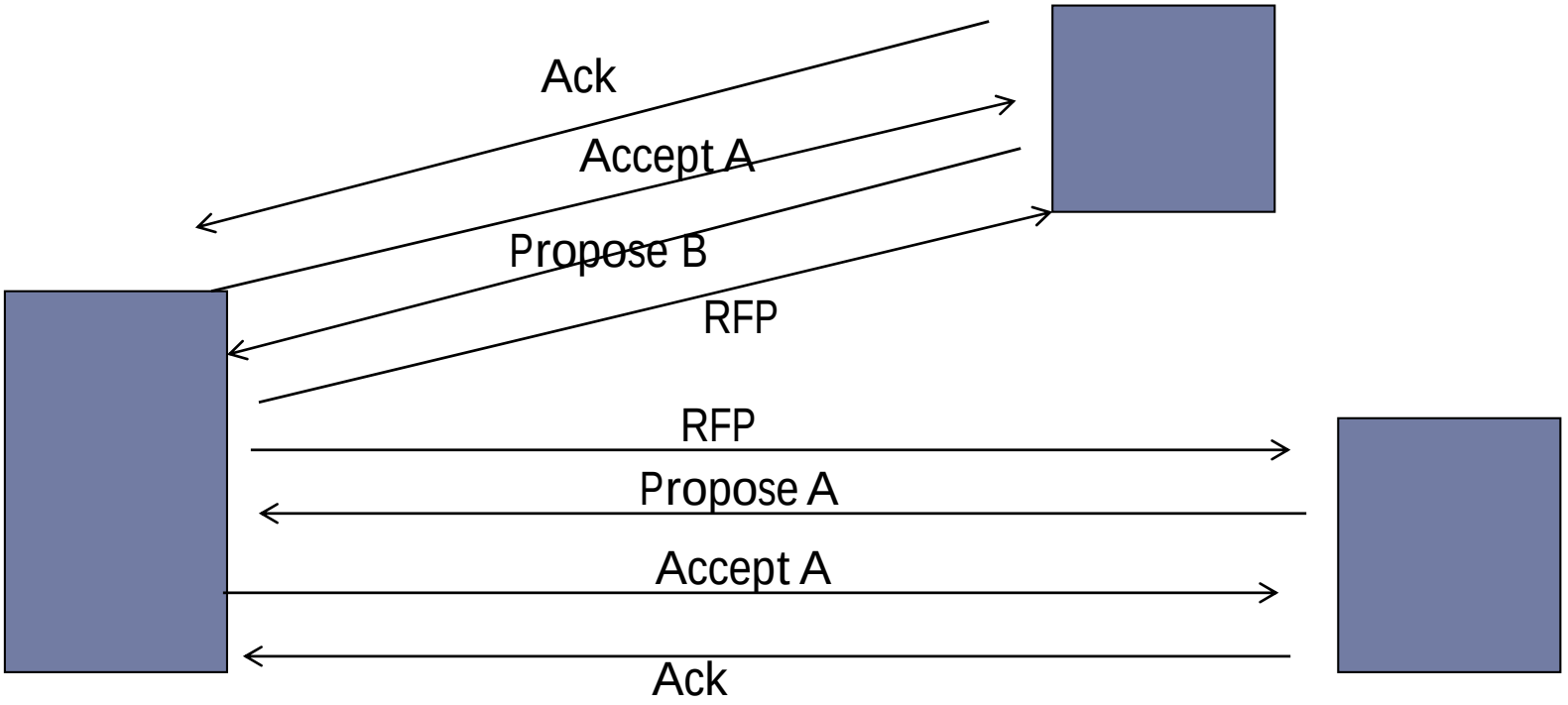
Communication Overhead:
Central processing needs to maintain complete model of world

Centralized Control: Advantages

- Simplicity of Coding
  - Because of
    - Single unified world view
    - Straightforward communication protocols
  - Resulting in
    - Cheaper development times
    - Easier maintenance and extensions
    - Enhanced security and predictability

Agent-based Systems: Problems

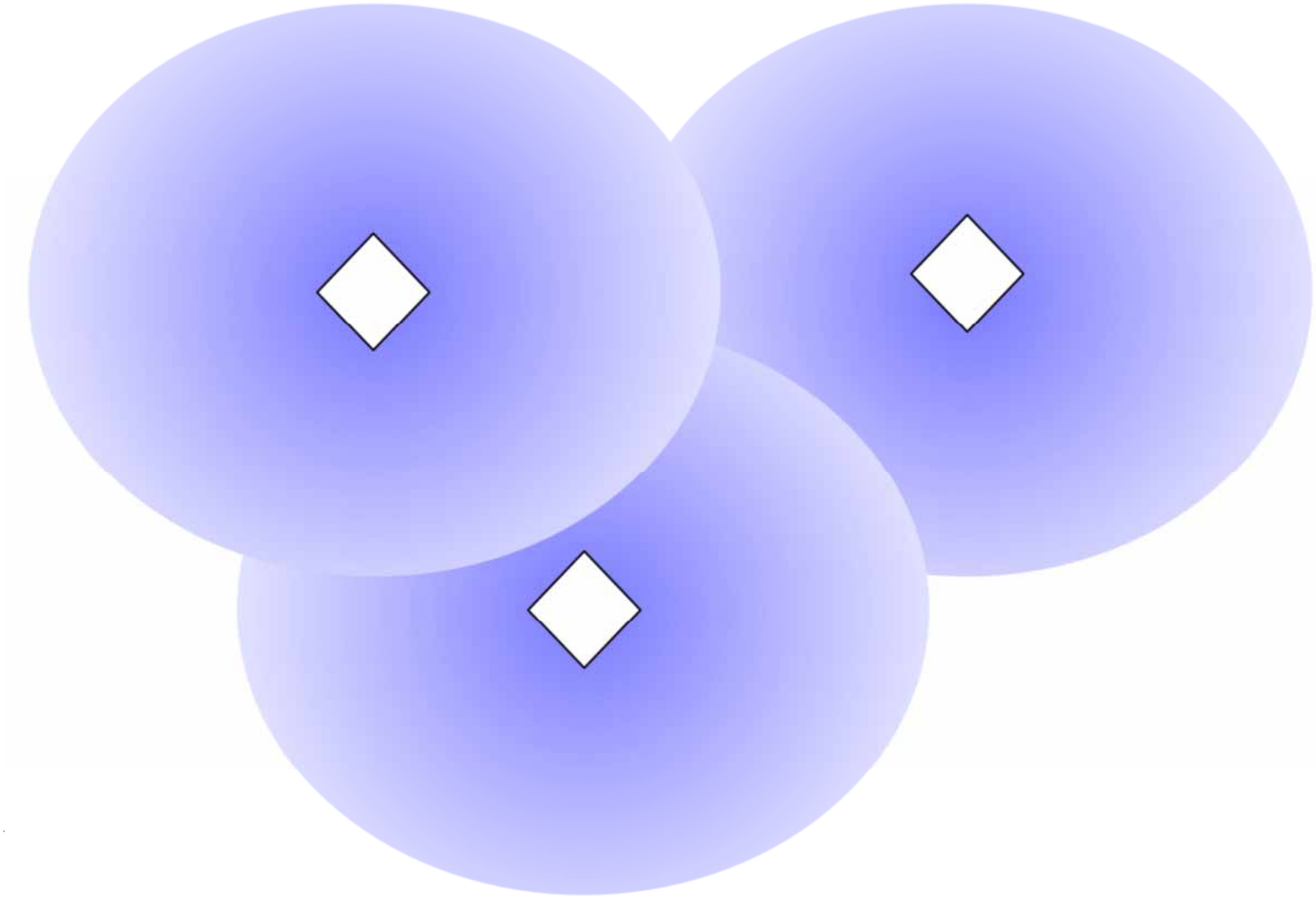
- Decision making is Complex:



Decision making in many negotiation protocols are NP complete!

Agent-based Systems: Problems

- No global view

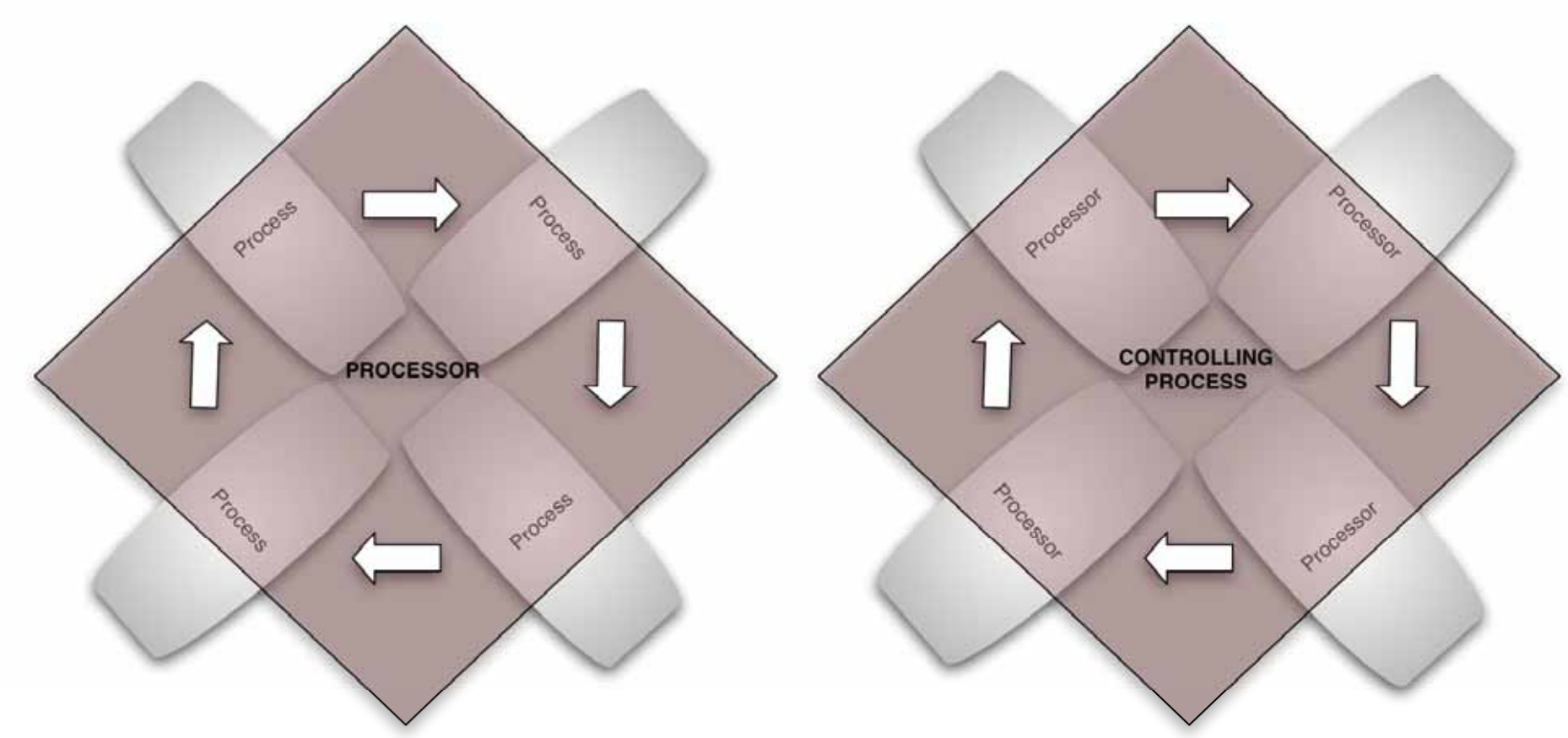


The Idea

A Process Integrated Mechanism …

- is a mechanism integrated at the *software level* rather than by physical contiguity of its parts
- has components that are conceived as parts of a single mechanism
  - even when they are physically separated and operate asynchronously.
- maintains a single unified world-view, and behavior is controlled by a single coordinating process.

PIM as “inverse time sharing”



A Time Sharing System

A Process Integrated Mechanism

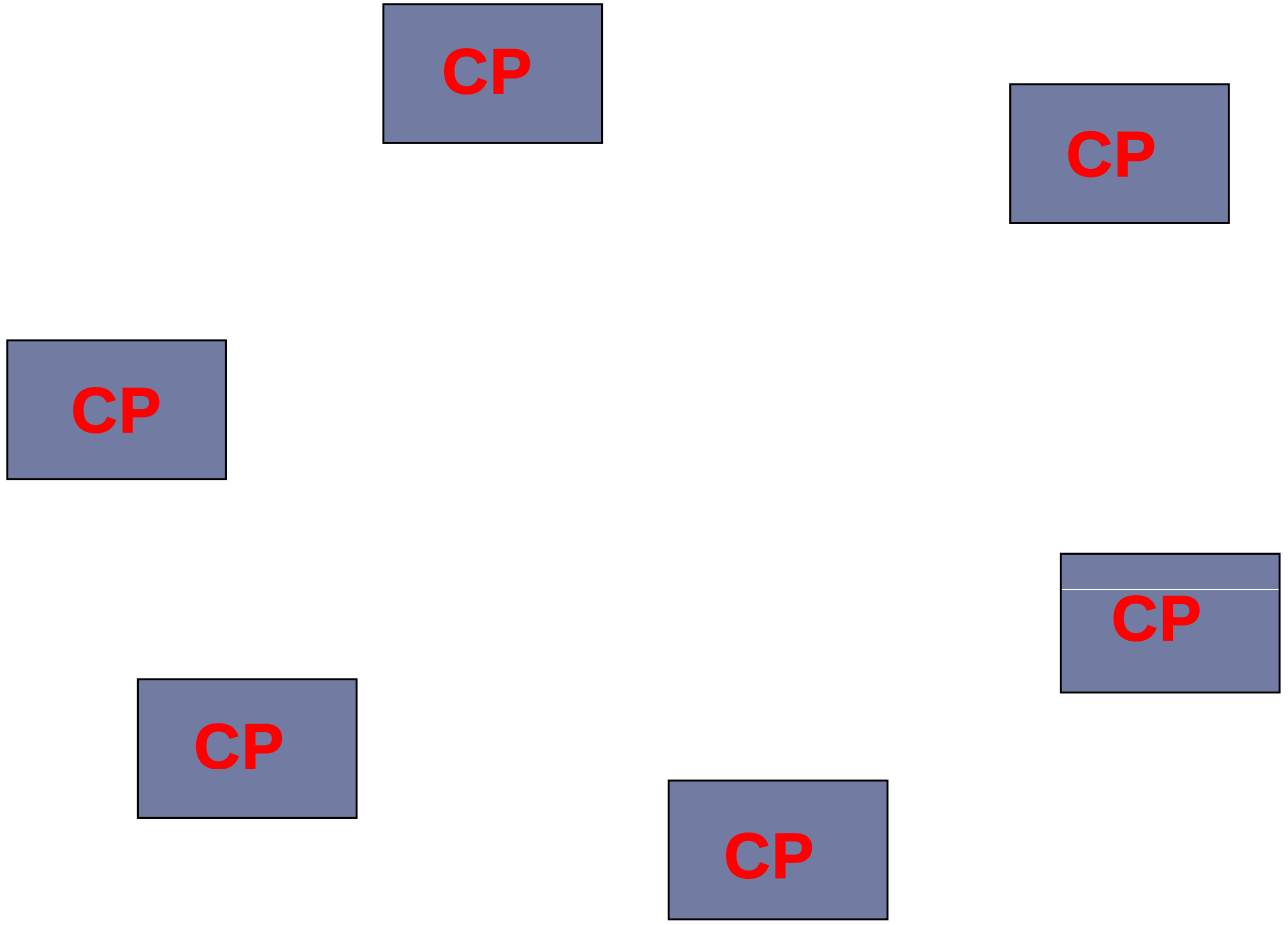
What was a PIM, again?

- There is one coordinating process (CP), running on exactly one component at any time
- Each component maintains a copy of the code for the CP - only the run-time state is moved
- The CP moves rapidly compared to the required reactivity of the system
- The movement of state is invisible to all, hidden in the runtime system
- The entire distributed system appears to the programmer as a single coherent platform

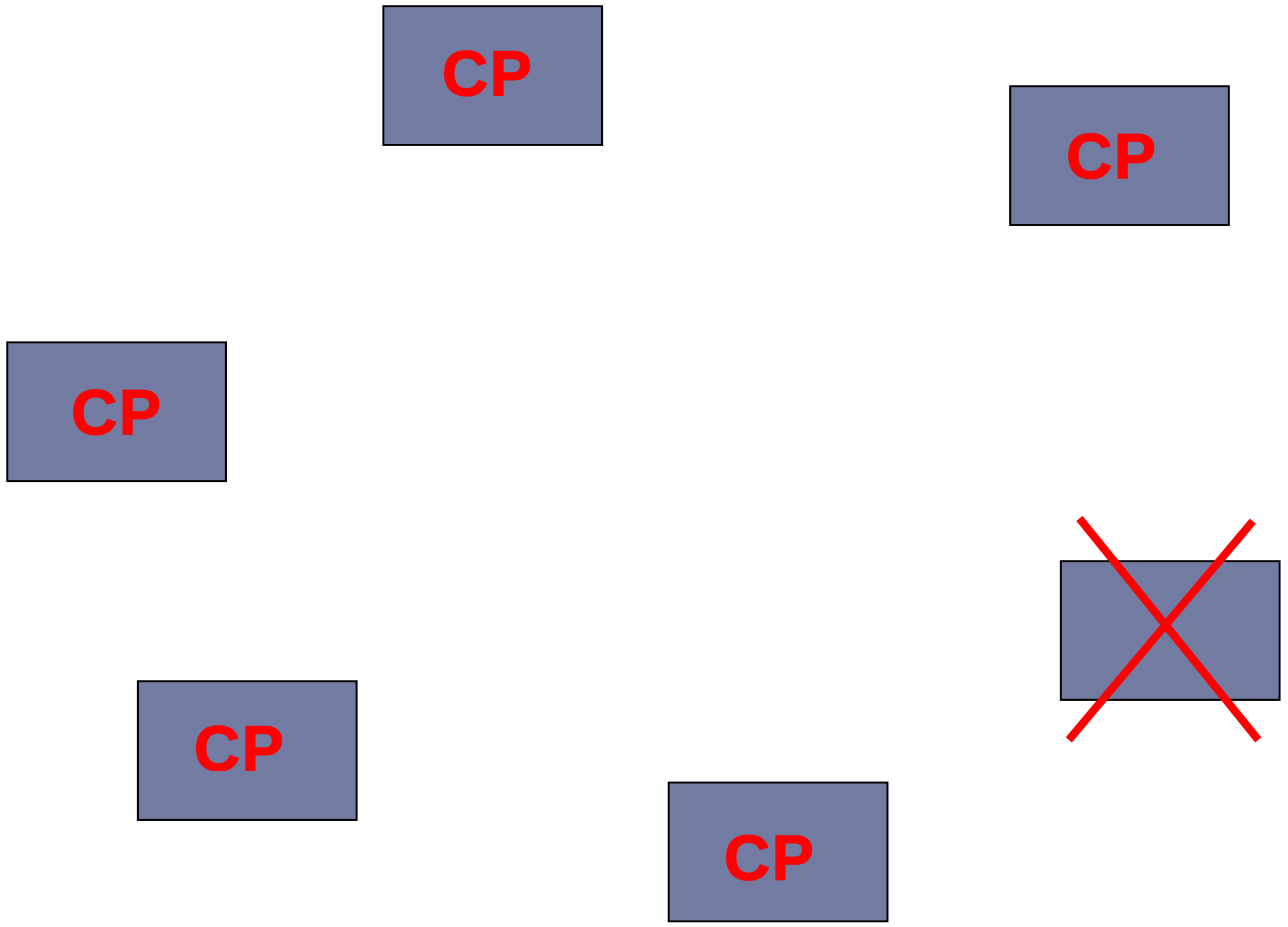
From Another Perspective...

- PIM raises the level of abstraction for the programmer
  - A distributed system appears to be and is programmed as a single, centralized system

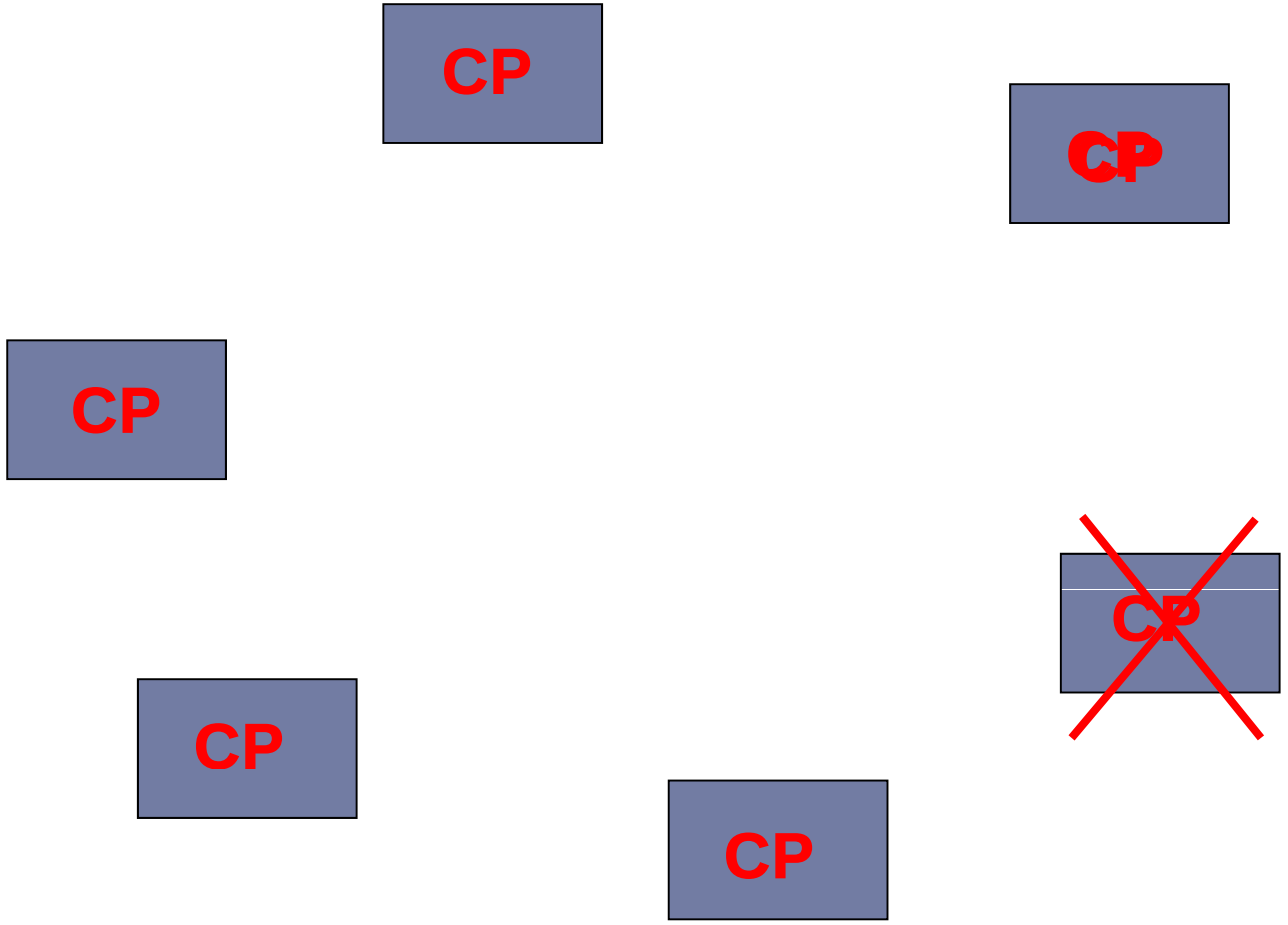
Robustness



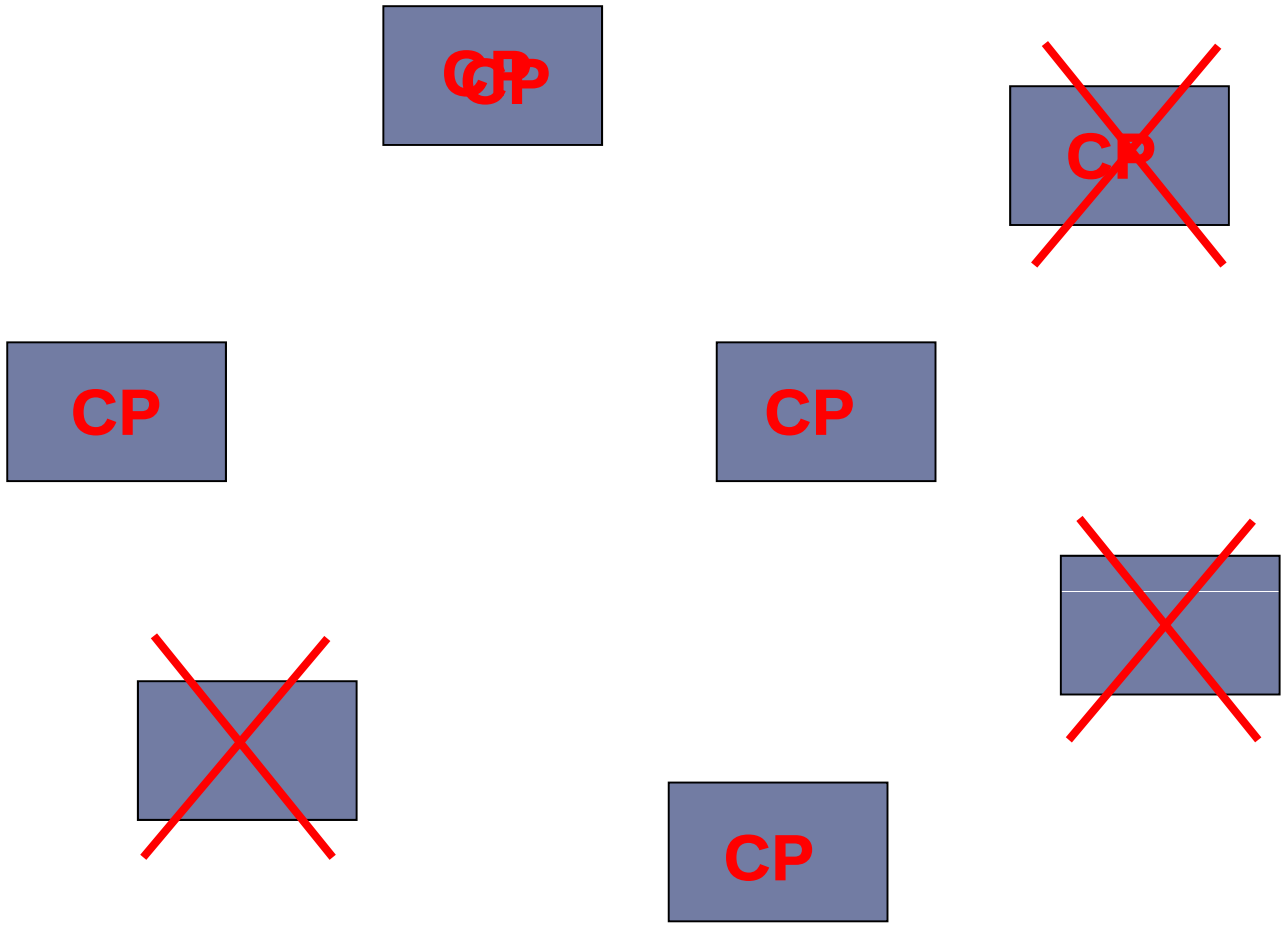
Robustness: Component Loss



CP component loss

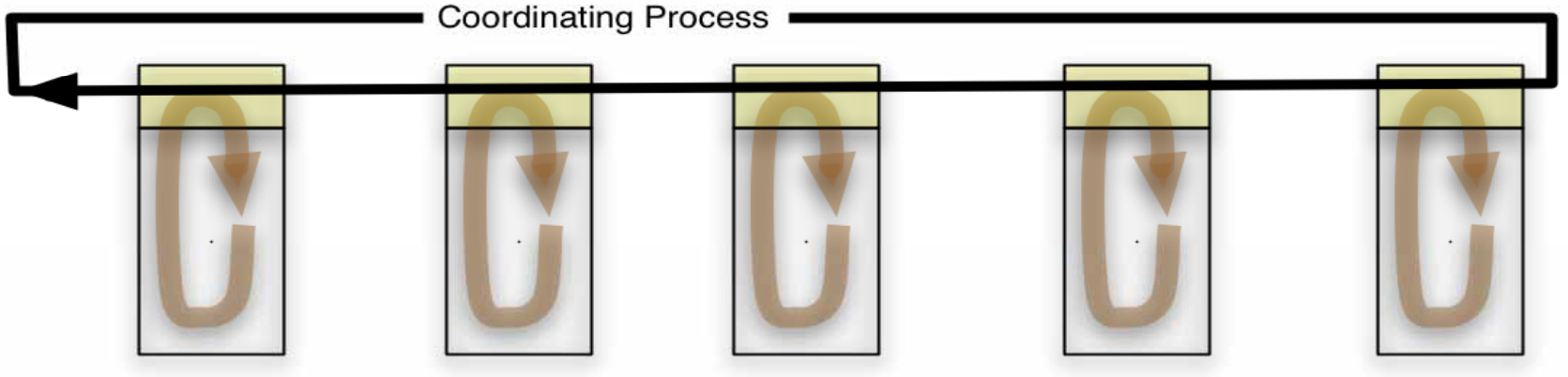


Inserting New Components



Coordinating vs Local Processes

- Each component may run local processes (e.g., sensor interpretation, effector control, …)
- Data is maintained locally, and is available to the coordinating process only when it is resident
- There is no explicit communication between components, information becomes known to all components only when it is "picked up" by the coordinating process



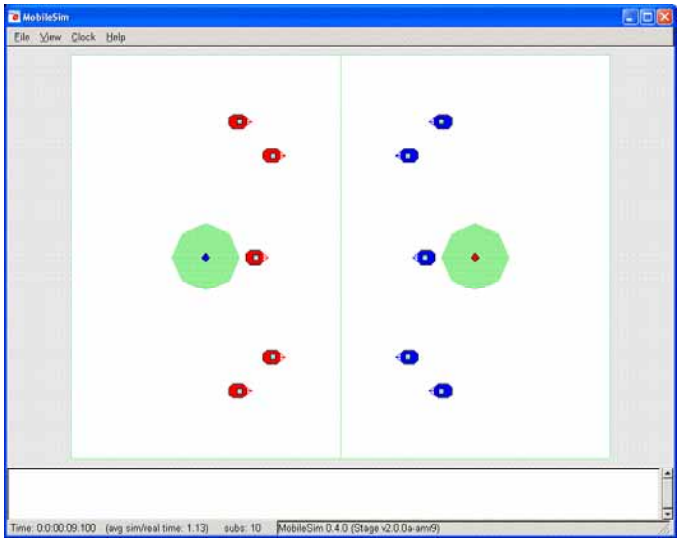
Some Advantages of PIMs

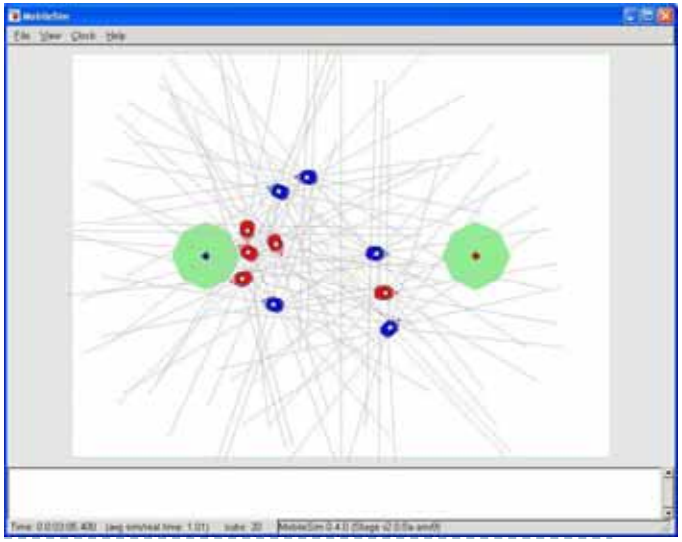
- Relative simplicity of programming
  - Programmed like a single agent with a global view
- Robustness to failure
  - There is no critical component, controlling process can be killed only by destroying all components
- Lower energy consumption
  - Reduced transmission requirements for data & control
- Safety
  - Shut down in one cycle of the controlling process
- Single point of contact to human team members

- The benefits of both centralized & distributed control without their problems

Some Initial Results

- Knock-Down-The-Flag Game
  - Played by two teams of robots – Agent-based team and PIM-based team
  - Analyze complexity of solution



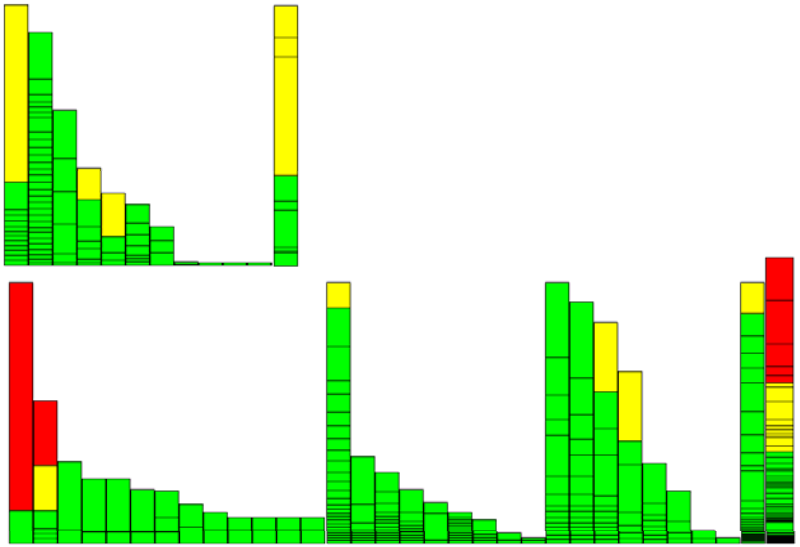


Complexity Results

- Analyze Lines of Code, Number of Classes, and Cyclomatic Complexity

Part	PIM		MAS	
	Lines	Classes	Lines	Classes
Coordination	671	11	4851	33
Role Assignment	538	3	1027	7
Path Planning	430	3	681	2

	Instruction count	Num of classes	Num of methods	Max CC	Methods CC > 10
PIM	2327	14	153	8	0
MAS	13428	40	337	55	13



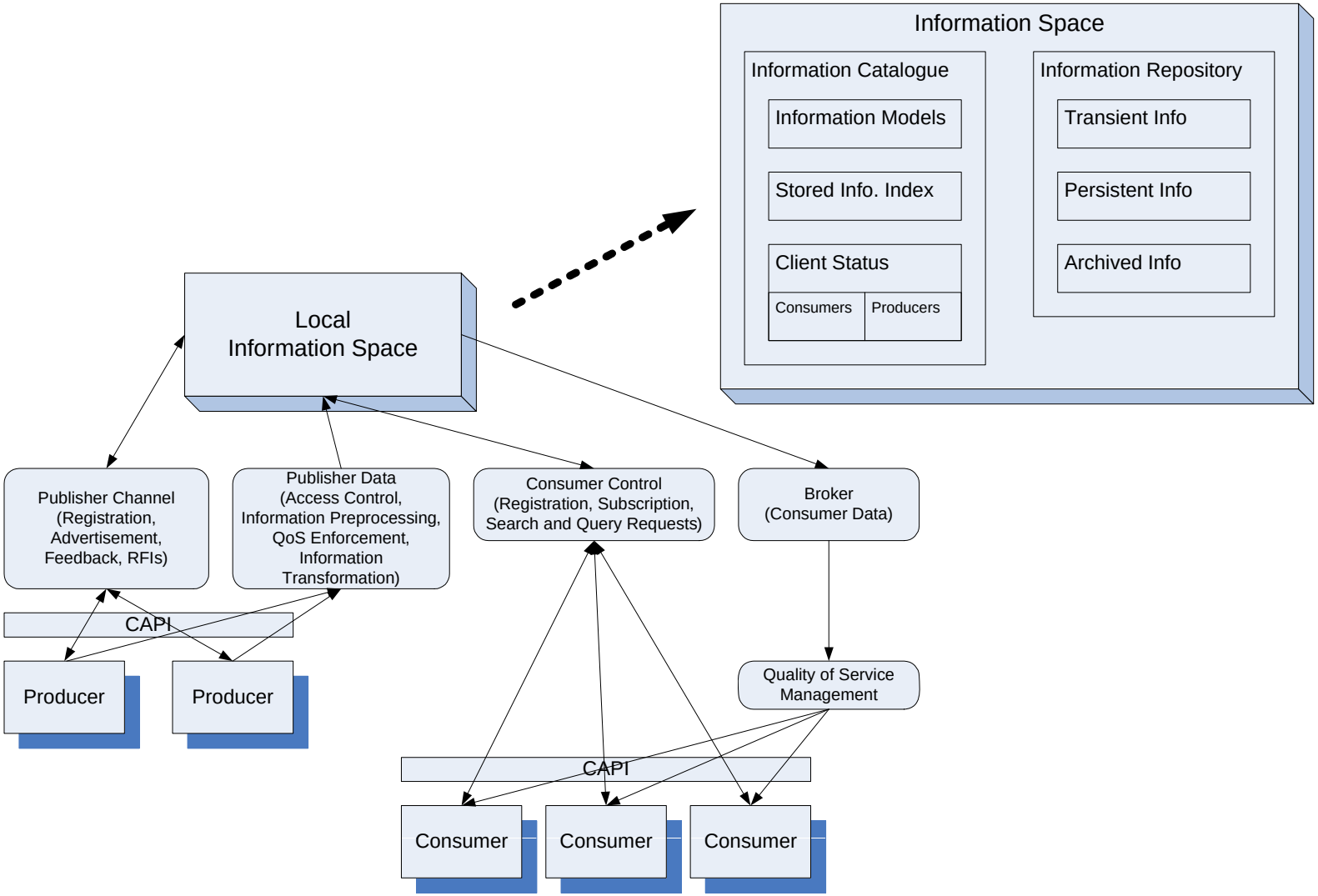
Top – PIM
Bottom – Multi-Agent System

Federated Information Spaces

Information Space (InfoSpace)

- A collection of clients exchanging information via an Information Management (IM) system
- IM System nominally consists of a server, a client interface (CAPI), and associated clients
- Information is packaged into Managed Information Objects (MIOs)
  - Consists of metadata (XML) and payload (binary)
- Client operations include
  - Publish
  - Subscribe
  - Query

Information Space (InfoSpace)



Vision for Federation

- Enable Interconnection Between Multiple InfoSpaces
- Interconnection is Peer-to-Peer
  - No master entity controlling federation
  - Federation is controlled independently from the perspective of each infospace
- Enable Sharing of MIOs Across InfoSpaces
  - Seamless subscriptions and queries across infospaces
  - Transparency to CAPI clients
    - Client-Server connections / communication untouched
  - Controlled via policies – not unrestricted
  - Identity and integrity of individual infospaces preserved
- Efficiency when Handling Subscriptions and Queries
  - Criteria: Latency, Bandwidth, Storage, Availability, Resource Utilization
- Policy-based Control over Federation

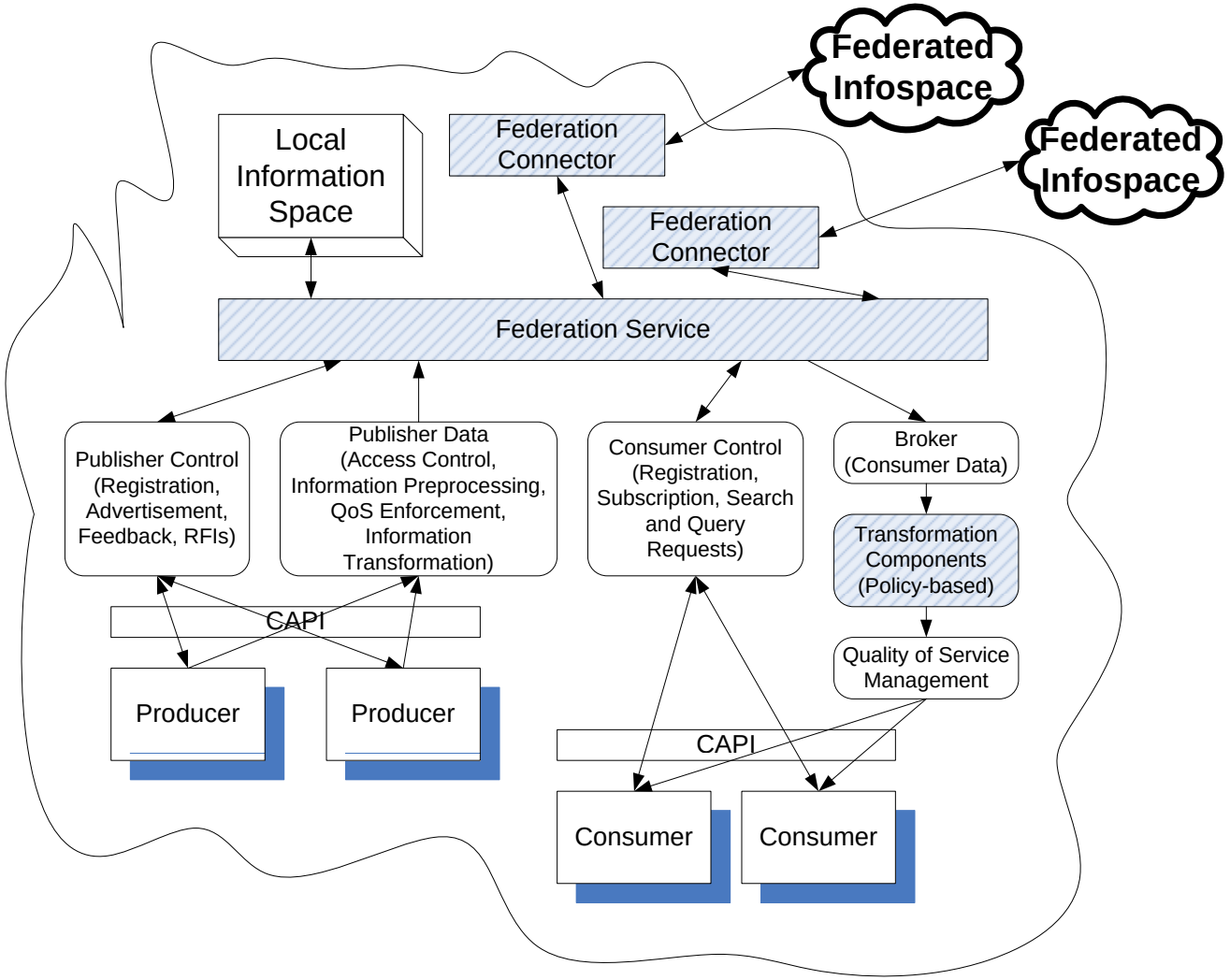
Research Goals

- Design Interface (API) for Federation
- Enable Federation Between Multiple Types of InfoSpaces
  - Enterprise – Enterprise
  - Tactical – Tactical
  - Enterprise – Tactical
  - Not limited to just two instances
- Control Federation Through Policies and Contracts
- Optimize Federation via Dynamic Adaptation
- Work with Multiple, Existing Implementations
  - Apollo (Done), Mercury (Done), Agile Computing
  - Phoenix (Abstract Architecture and Fawkes)

Federation and Coalitions

- Federation is a natural model to support coalitions
  - Information spaces are independently managed
- Interconnection is peer-to-peer
- Federation is independently controlled by each participant
- KAoS Policies regulate
  - Formation of federation
  - Nature and type of information exchanged
    - Both metadata and data
  - Nature and type of queries supported
  - Resources contributed to federation

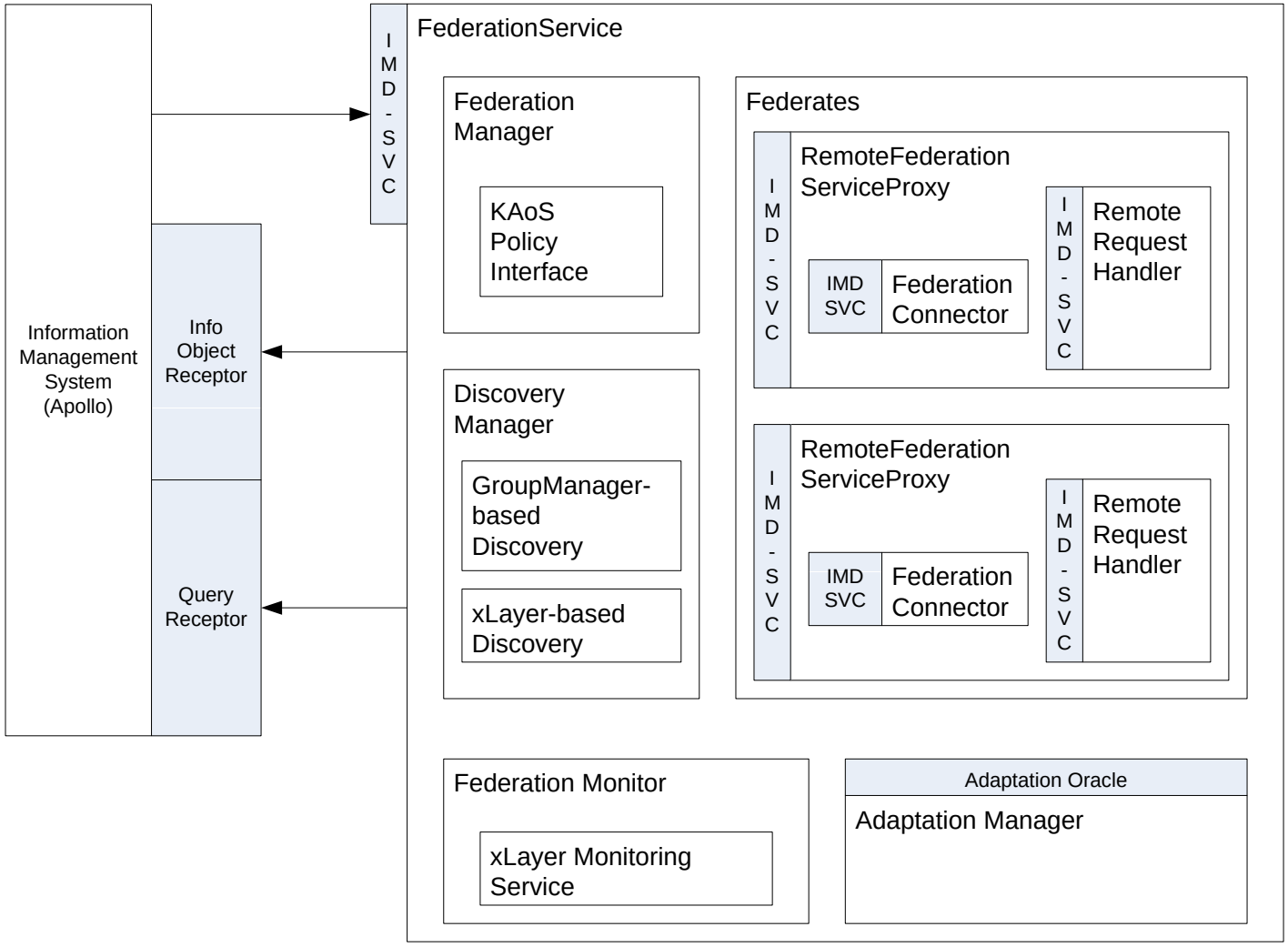
InfoSpace Architecture with Federation Service Components



Major Components

- Federation Service
  - Main component – contains everything else
  - Infospace interacts only with this component
- Federation Manager
  - Controls whether federation happens when a remote infospace is discovered
  - Negotiates trust and federation contract
- Federation Connector
  - Handles the wire protocol between two federated infospaces
- Remote Federation Service Proxy
  - Stores remote predicates and matches locally published MIOs
  - Policy enforcement point for federation policies
- Discovery Manager
  - Handles discovery of other infospaces potential federate
- Federation Monitor
  - Monitors and visualizes behavior and performance of federation
- Federation Adapter
  - Dynamically optimizes the activities of federation

Major Components

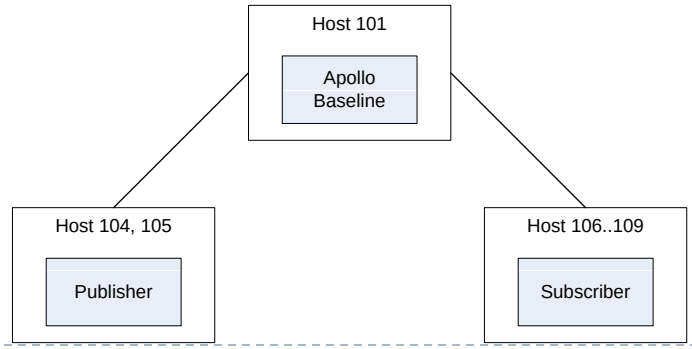


Evaluation Setup (1)

- Measure Overhead from Inserting Federation Service
  - Performance of pub/sub/query with and without Federation Service in Apollo
- Performance of pub/sub/query between locally attached clients versus clients attached to federates
- Tests use the standard benchmark test included with the Apollo distribution
  - mil.af.rl.im.benchmark
- Test Descriptions
  - Developed scripts and configurations for 12 benchmark test suites
  - The test were run on the IHMC MLAB Testbed
    - Homogenous network of 16 nodes
    - Specifications: Intel Celeron Processor at 2.66 GHz with 1 GB RAM, interconnected with 100 Mbps Fast Ethernet, running the Linux operating system

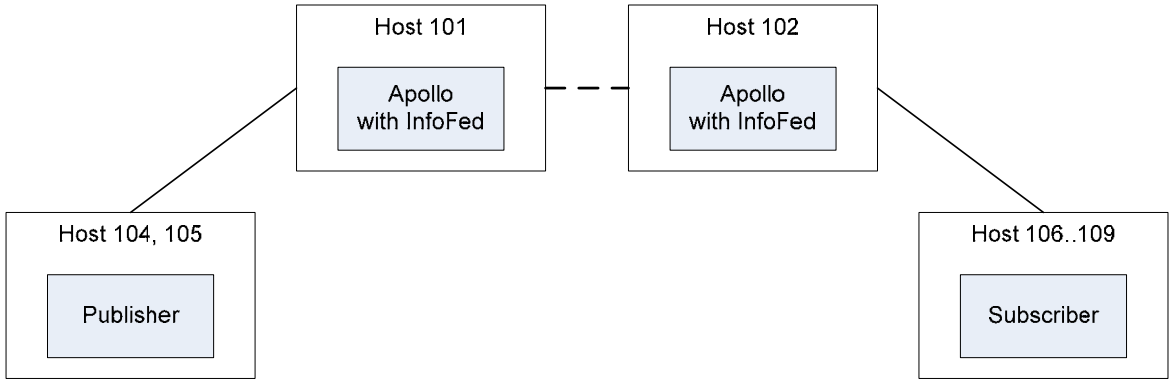
Evaluation Setup (2)

- Independent (Control) Variables:
  - Baseline Apollo and Apollo with Federation
  - Number of Publishers, Subscribers, and Federates
  - Payload Size
- Dependent Variables:
  - Performance (execution time) of Publisher and Subscriber(s)
  - Latency of Information Delivery
- Baseline Configuration for Throughput

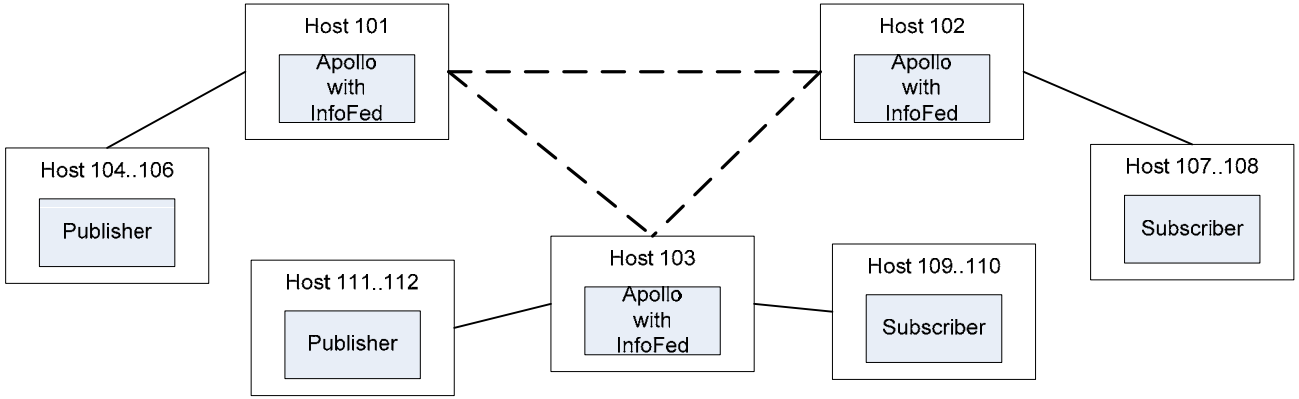


Evaluation Setup (3)

- Two Federate Configuration for Throughput

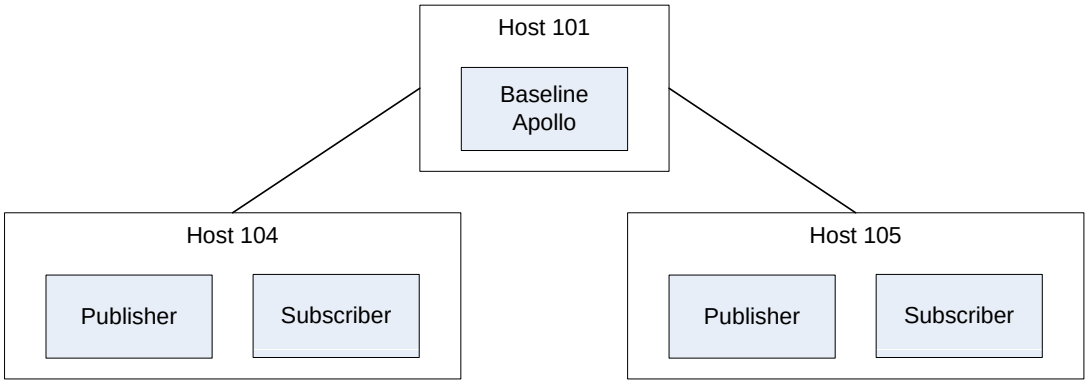


- Three Federate Configuration for Throughput

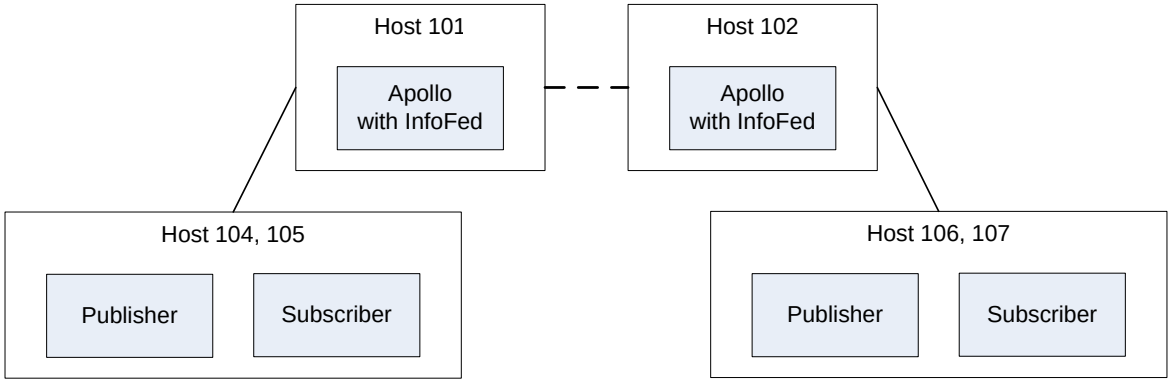


Evaluation Setup (4)

- Baseline Configuration for Latency



- Two Federate Configuration for Latency



Results (1) – Throughput

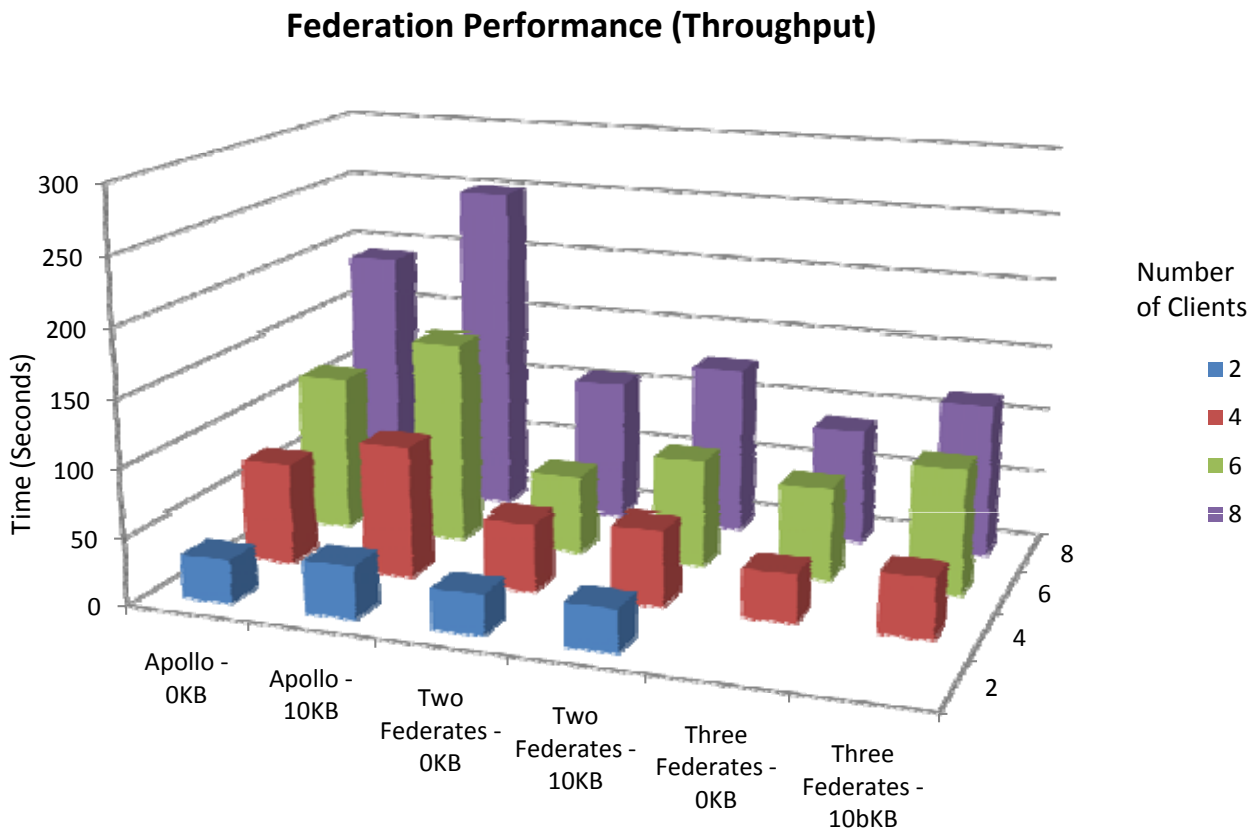
- Publisher Performance

Publisher Performance (Time in Seconds to Publish 1275 Objects)						
# Clients	Apollo - 0KB	Apollo - 10KB	Two Federates - 0KB	Two Federates - 10KB	Three Federates - 0KB	Three Federates - 10bKB
2	32.04	38.87	29.85	31.12		
4	74.25	97.1	50.52	56.1	35.88	44.3
6	116.94	152.25	57.31	79.22	67.86	92.38
8	191.67	250.18	106.87	125.99	86.4	114.91

Publisher Performance (Improvement!)				
# Clients	Two Federates - 0KB	Two Federates - 10KB	Three Federates - 0KB	Three Federates - 10bKB
2	1.07	1.25		
4	1.47	1.73	2.07	2.19
6	2.04	1.92	1.72	1.65
8	1.79	1.99	2.22	2.18

Results (2) – Throughput

- Publisher Performance



Results (3) - Throughput

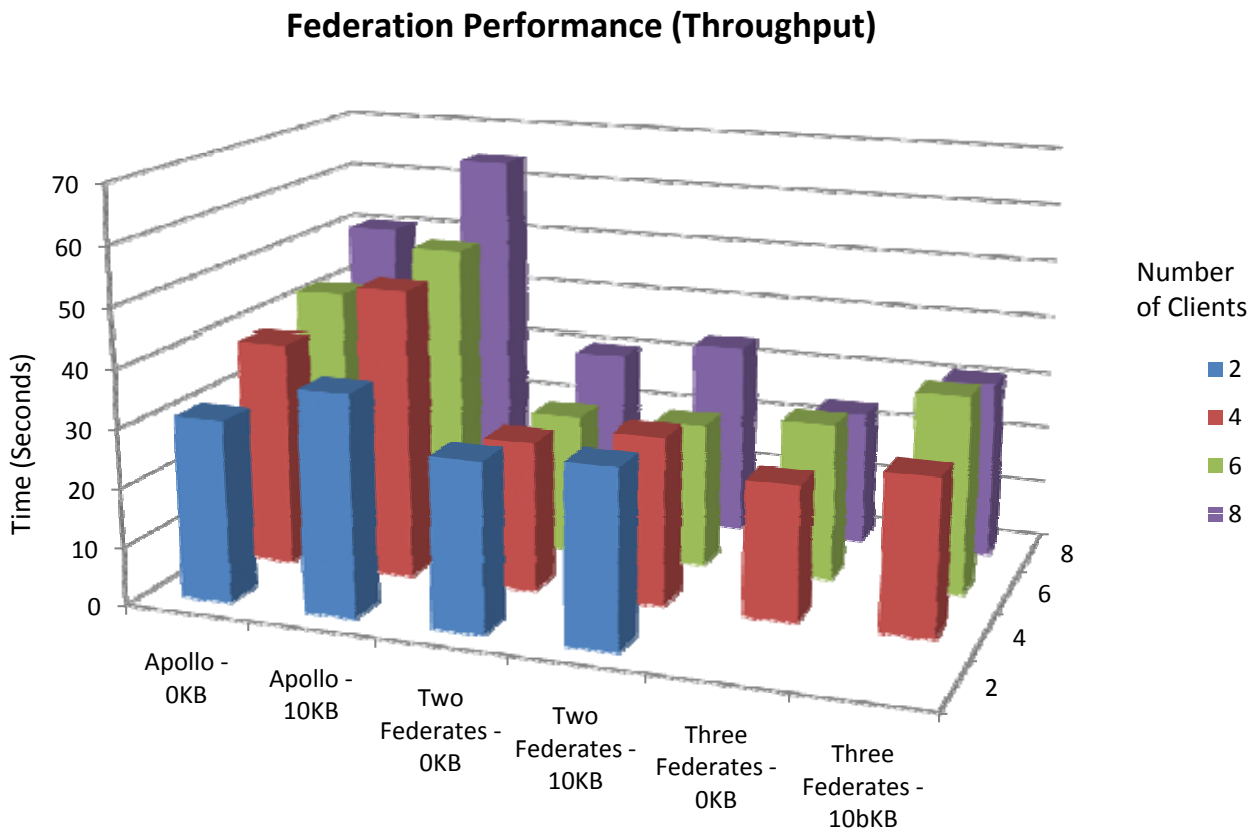
- Subscriber Performance

Subscriber Performance (Time in Seconds to Receive 1275 Objects)						
# Clients	Apollo - 0KB	Apollo - 10KB	Two Federates - 0KB	Two Federates - 10KB	Three Federates - 0KB	Three Federates - 10bKB
2	31.08	37.65	28.79	30.11		
4	38.76	49.43	25.78	28.8	23.16	26.83
6	42.71	51.98	24.38	24.9	27.13	34.13
8	50.42	64.18	30.28	33.56	23.29	30.84

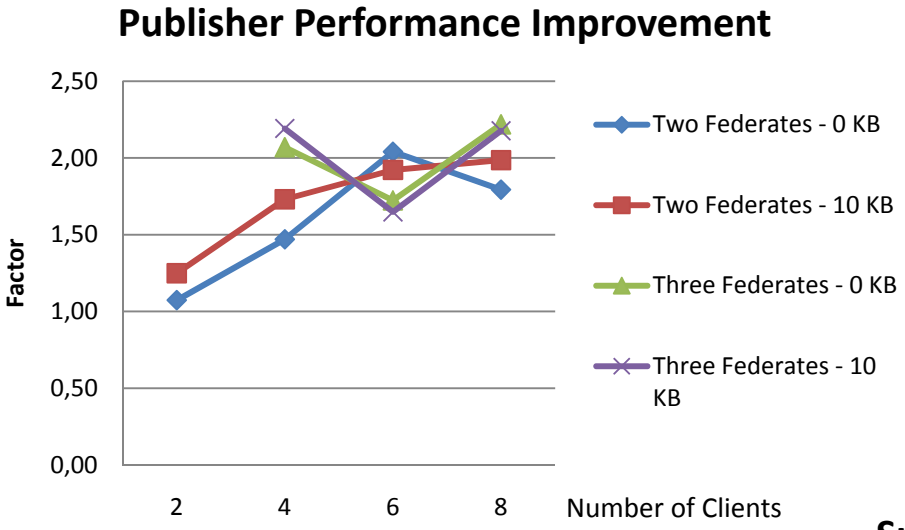
Subscriber Performance (Improvement!)				
# Clients	Two Federates - 0KB	Two Federates - 10KB	Three Federates - 0KB	Three Federates - 10bKB
2	1.08	1.25		
4	1.50	1.72	1.67	1.84
6	1.75	2.09	1.57	1.52
8	1.67	1.91	2.16	2.08

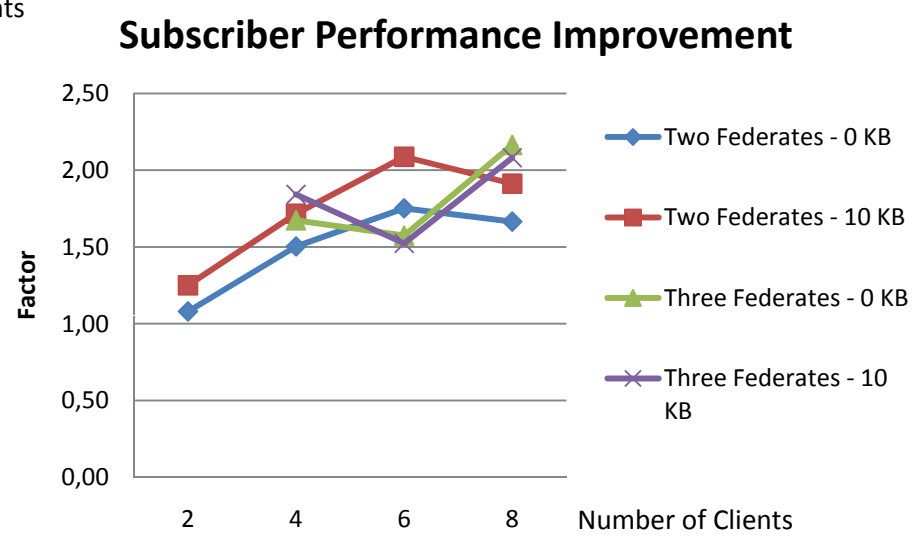
Results (4) – Throughput

- Subscriber Performance



Results (5) - Throughput

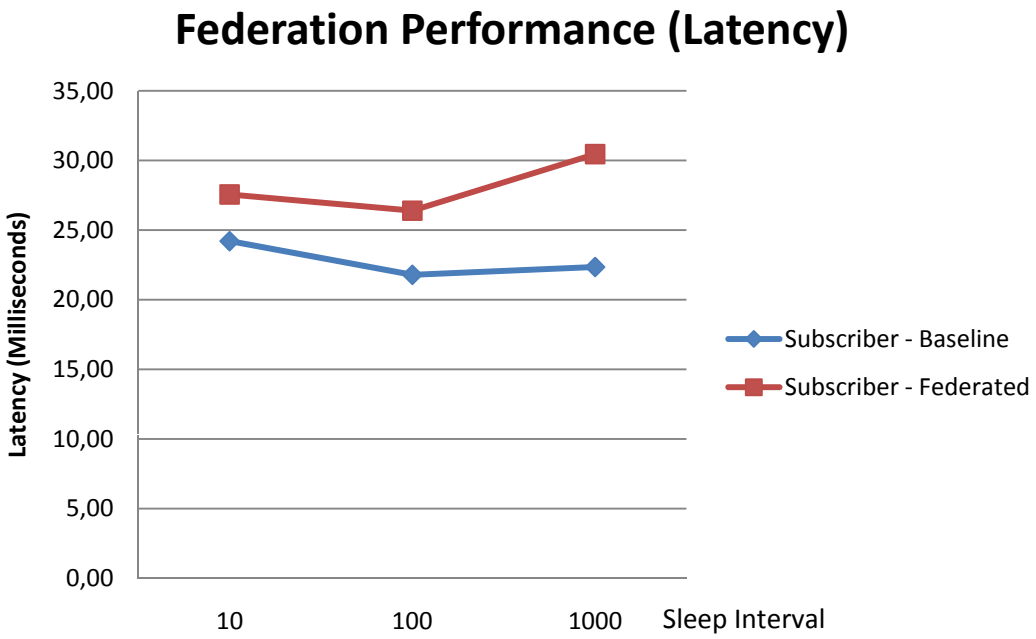




Results (6) - Latency

- Publisher and Subscriber Latency

Subscriber - Latency			
Sleep Time (ms)	10	100	1000
Publisher - Baseline	41.61	154.58	1302.58
Publisher - Federated	39.44	153.69	1302.04
Subscriber - Baseline	24.21	21.79	22.35
Subscriber - Federated	27.56	26.40	30.46



Results (6) - Query

Performance of Standard Query Example

- Queries Utilized (From Apollo Test Suite):
  starts-with(/theSigma/theData/sigmaOneThousandsNots, "sigmaOneThousandsN")
  not(/theSigma/theData/sigmaOneThousandsNots="sigmaOneThousandsOKs")
- Database populated with 1000 objects

Configuration	Performance (All Times in Seconds)								
	No Subscribers	One Subscriber		Two Subscribers		Three Subscribers		Query	
	Publisher Time	Pub	Sub	Pub	Sub	Pub	Sub	Ex 1	Ex 2
Local Performance of Baseline Apollo	198		238		311		363	8	47
Local Performance of Apollo Patched with Federation	197		246		319		432	8	51
Performance With Federation	208		221		223		251	56	99

(Will be Updated Shortly for Three Federates)

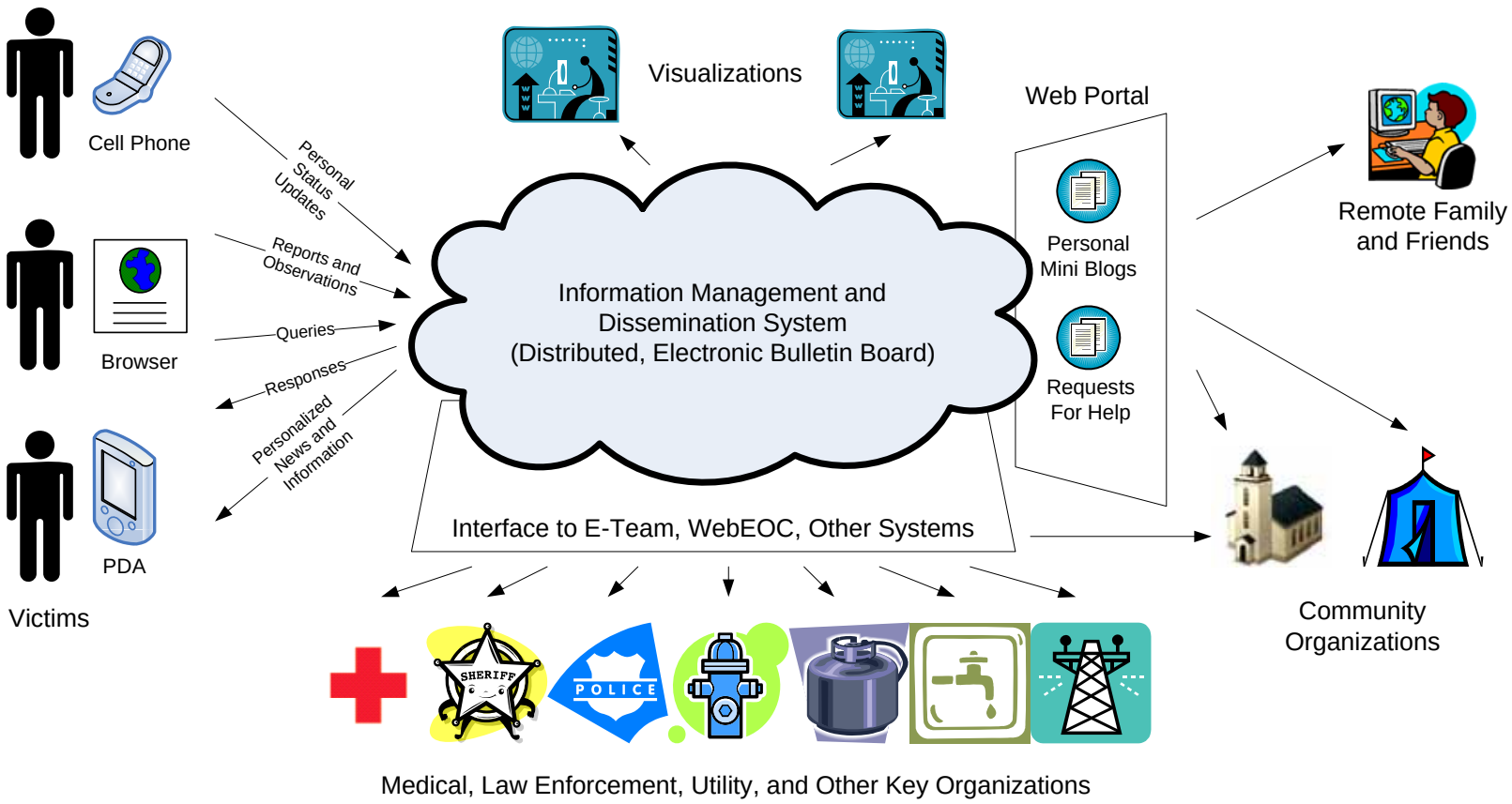
Work in Progress / Future Ideas

Disaster Area Information Management System (DAIMS)

Goals

- Disasters cause significant disruption in infrastructure
  - Example: Katrina
- Leverage and extend current work to disaster recovery
  - Capabilities in Agile Computing and Federated Information Spaces
- Enable victims to
  - Use existing hardware – cell phones (ubiquitous) and simple interfaces
  - Get information out about themselves
  - Contribute to the recovery effort – make them into "sensors"
  - Obtain status and information they need
- Integrate organizations and manage information flow between organizations, victims, and other users

System and Concept of Operations



User Perspective

- User's already have cell phones – but they will not work
- Deploy an ad-hoc infrastructure for to enable cell phones
  - Nano cells, femto cells, etc.
  - No voice calls (too much bandwidth) – just data access
- Users can use SMS, WAP, WWW, (or voice-based interfaces) to interact with system
- Publish information about themselves
  - Their status, health, location
  - Essentially – create a small "page" or blog about themselves
  - Perhaps even extend it to contain a sequence of updates – a blog (like other social-networking pages)
- Publishing is important – one-to-one communication too expensive
  - Plus you never remember all the people you need to inform!
- Who can obtain the information?
  - Anyone in the world – through the Web – provided they are "authorized"
  - Need creative security mechanisms
  - One solution – person looking up a victim must know some subset of the information (telephone number, name, address, DoB, etc.)

Turning Users into Sensors

- Users should be allowed to report observations / incidents
  - Reports about downed power line, fire, vandalism, gas leak, water leak, etc.
  - Information on availability (food, water, gas, services, etc) or store openings
- User Interface is Important
  - Simple WAP drill-down interface
  - SMS or voice-based (natural language with limited vocabulary)
  - System can use triangulation to estimate position
- Reports are routed to appropriate agencies as necessary
- Information is made available to users who need supplies, services, etc.

Other Challenges

- Trust and Reputation Services
  - Discount information from unreliable users
- Information Pedigree
  - Track source of any information injected into the system
- Information Consolidation
  - Ten reports about a downed power line in the same vicinity should become one report
- Policy-based Control Over Information Exchange / Release
  - Don't release information that cannot be shared between agencies
  - Don't release information that will cause additional problems
    - One location reporting availability of gas might cause everyone to try and go there
    - Perhaps wait until multiple locations are available?
- Resource allocation
  - Bandwidth and power are constrained
  - Need to prioritize traffic on the network

Information Oriented Networking (ION)

Information Oriented Networking (ION)

- Raise the Level of Abstraction for Users of Networks
  - Google has already done this in some ways for the Web
- Goals
  - Users specify information needs – let system determine ways to obtain it
    - Users should not have to specify sources, workflows, fusion processes
  - Users should not have to worry about QoS, prioritization, security
  - ION should provide meta-information about the quality of the information
    - Level of trust, accuracy, timeliness, pedigree
  - ION should be proactive and push information, based on user context

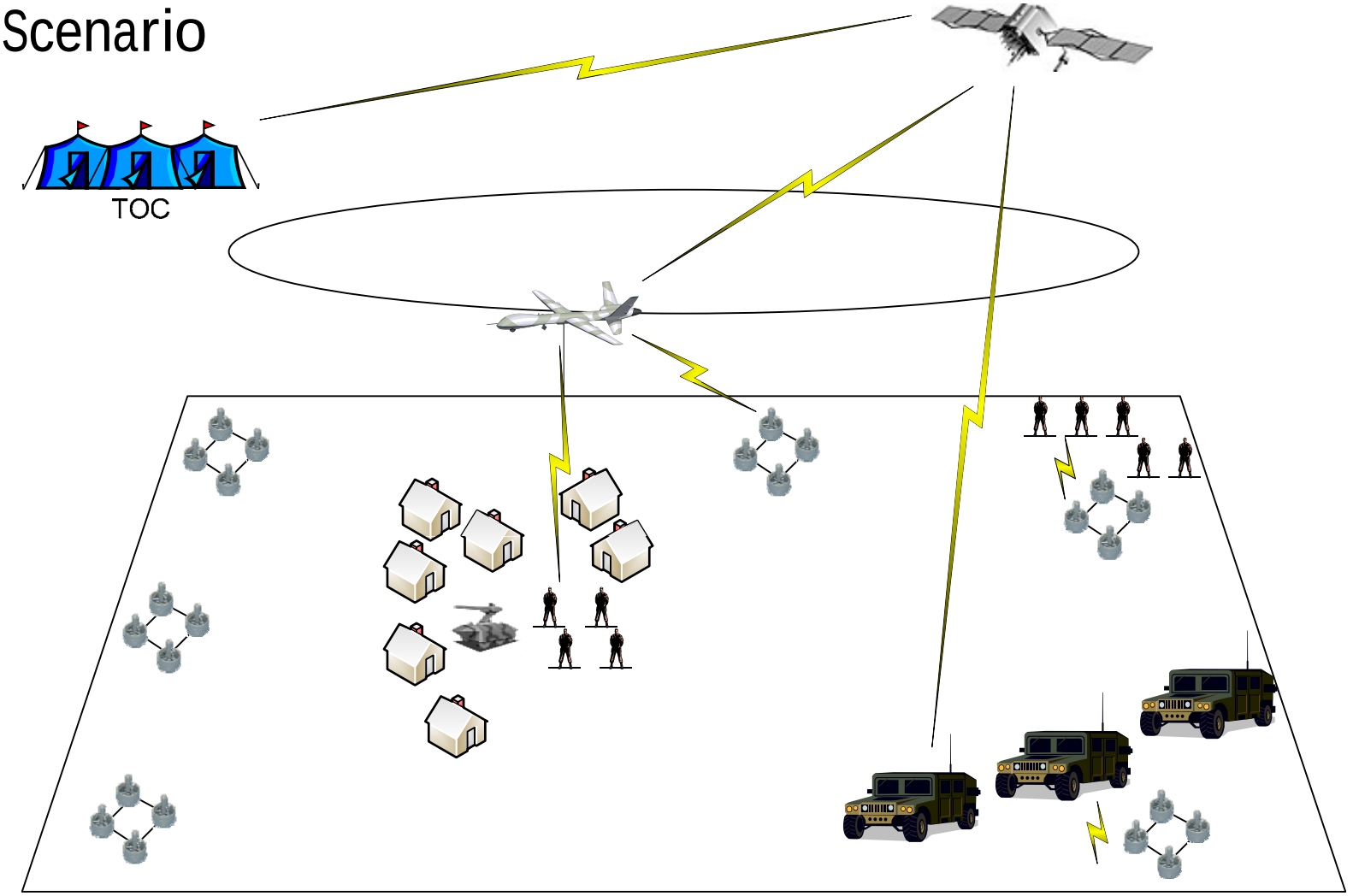
Network Science

Network Science – What is it?

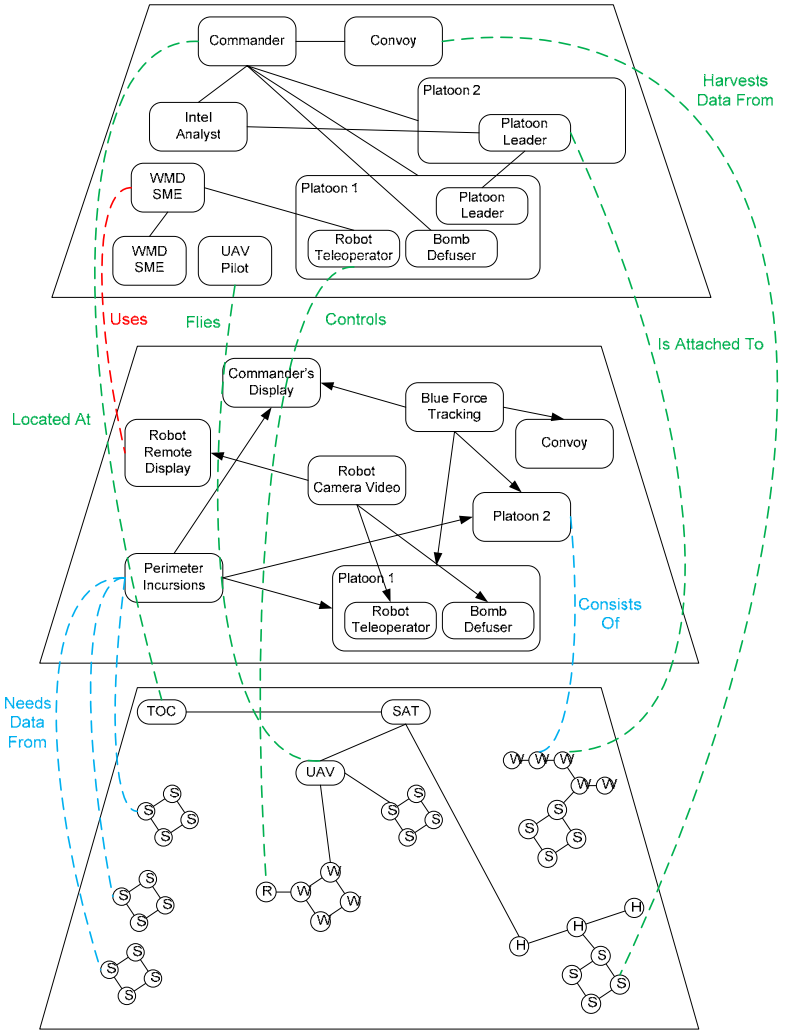
- Two Viewpoints
  - Study of networks by modeling them as graphs and applying statistical analysis
    - Networks may be of multiple types
    - Major results:
      - Small-world properties, scale-free (as opposed to random) networks
  - Integrate multiple types of networks to build adaptive systems
    - Communication networks
    - “Information” networks
    - Social and Cognitive Networks

Example of Integrated Network

- Scenario



Model of Integrated Network



Optimization Across Networks